# Recent-Secure Authentication: Enforcing Revocation in Distributed Systems

Stuart G. Stubblebine

AT&T Bell Laboratories
101 Crawfords Corner Rd., Holmdel, NJ 07733-3030
stubblebine@research.att.com

## Abstract

*A general method is described for formally specifying and reasoning about distributed systems with any desired degree of immediacy for revoking authentication. To effect revocation, 'authenticating entities' impose freshness constraints on credentials or authenticated statements made by trusted intermediaries. If fresh statements are not presented, then the authentication is questionable. Freshness constraints are derived from initial policy assumptions and authentic statements made by trusted intermediaries. By adjusting freshness constraints, the delay for certain revocation can be arbitrarily bounded. We illustrate how the inclusion of freshness policies within certificates enables the design of a secure and highly available revocation service. We illustrate the application of the method and new techniques in an example.*

## 1. INTRODUCTION

An authentication architecture that scales globally is desirable to support authentication and authorization for electronic commerce. A characteristic of universal electronic commerce is that clients and commercial servers not previously known to one another must interact. An essential feature of an authentication service in large distributed systems is revocation. *Revocation* entails rescinding authentication and authorization statements that have become invalid. Revocation is needed because authentication information changes with time, perhaps, due to a compromise or suspected compromise of an entity's private key, change of affiliation, or cessation of an entity's operation. When a compromise is discovered, rapid revocation of information is required to prevent unauthorized use of resources and electronic fraud.

Understanding revocation is an important business concern to service providers as well as to users of an authentication service; it has been estimated that the yearly running expenses of an authentication infrastructure derive mainly from administrating revocation [3].

A service for revoking authentication should have the following desirable properties:

- *Definite*. Revocation should be fail-safe or assured with bounded delays.
- *Available and Recent*. The mechanisms for posting and retrieving updates must be highly available and retrieved information should be recent (if not current).
- *Adjustable.* Protection and performance trade-off should be adjustable to suit varying risk adversity.
- *Bounded Recovery.* When a compromise is discovered, delays in revocation should be decidedly bounded in time.
- *Contained.* A compromised revocation service should not allow illegitimate identification credentials to be issued.

Factors inherent to the difficulty of revocation in a large distributed environment include:

- *Size.* Numerous entities not previously known to one another may need to interact securely.
- *Trust.* Entities have different views of the trustworthiness of intermediaries and of each other.
- *Security.* Protection of computing environments are variable and uncertain.
- *Distributed and Faulty.* The authenticating entity's knowledge of authentication information can be inaccurate due to communication latency, failures, and active wiretapping.
- *Temporal Dynamics.* Authentication and authorization information changes with time and is unpredictable.

To date, little has been published focusing on revocation in large distributed systems, perhaps, because few large scale authentication architectures exist that cross multiple autonomous realms. Network authentication services, such as Kerberos [18] and DCE [21] (which is built on Kerberos), have seen reasonable deployment within local autonomous realms. However, experience with inter-realm authentication is somewhat limited by the current dearth of inter-realm applications. Authentication services based on shared-secret cryptosystems (e.g., DES [17]), have inherent drawbacks to scaling to large distributed systems. In particular, authentication requires trusted on-line interaction potentially with numerous intermediate servers for each initial authentication. Also, they require on-line servers to maintain the confidentiality of shared secrets. The compromise of any intermediate server can lead to security exposures beyond the boundary of the compromised realm[9].

Authentication in large distributed systems is moving towards the integration of local network authentication servers with global directories (e.g., those based on the X.500 directory [28]) and open authentication architectures (e.g., X.509 [29]) using public key cryptography [7,20,26,29]. Public key crytposystems like RSA [22] and the Digital Signature Standard (DSS) [6] have a private key that is kept secret and a public key that can be published to others.

Global authentication architectures based on public key cryptography assume that named principals to be authenticated maintain the confidentiality of their private keys. Certificates using public key cryptography enable authentication information to be distributed using servers that need not be trusted on the authority of the certificate contents. Intermediaries, called certifiers or *certification authorities* (when authority is assumed by authenticating entities), create cryptographically protected statements called *certificates*. Identification authorities, having authority on identification of entities, issue *identification certificates*. Identification certificates assert that a public key is associated with an entity having a unique name. *Revocation authorities*, having authority on the status of certificates, issue revocation certificates. *Revocation certificates* assert the status of certificates previously issued. *Revocation policies*, which are typically included in authentication policies, represent the bounded delay before an authentication entity becomes current on the accuracy of authentication information. Authentication conforming to these policies is called *recent-secure authentication*. The entity or agent doing the authentication is called the *authenticating entity.*

Propagation of certificates through servers, such as directory servers, can introduce delays. In this paper, we formalize data-origin authentication policies that enable authentication to be as strong as a one-way authentication using timestamps or arbitrarily weaker.[1]

To better understand the protection offered by authentication, authorization, and revocation in distributed systems, and to improve these systems, we present a general method for formally specifying and reasoning about distributed system with any desired degree of immediacy for revoking authentication. To effect revocation, authenticating entities impose freshness constraints on statements made by trusted intermediaries. These constraints are derived from initial policy assumptions and authentic statements made by trusted intermediaries. By adjusting freshness constraints the delay for certain revocation can be arbitrarily bounded. Though our techniques apply to authorization built on an authentication platform, the focus of this paper is authentication.

The remainder of the paper is organized follows. In Section 2, an intuitive justification for the utility of recent-secure authentication is presented. In Section 3, the specification language and axioms for reasoning about recent-secure authentication are presented. In Section 4, recent-secure authentication is defined and revocation mechanisms are discussed. In Section 5, we present a technique for delegating revocation authority yet retaining both identification authority and authority for specifying recent-secure authentication policies. We explain how this technique enables the design of secure and available revocation services. In Section 6, we illustrated our method by analyzing an authentication instance of an experimental authentication architecture. In Section 7 we present our conclusions of this paper.

## 2. BACKGROUND

Recent-secure authentication is based on the specification of *freshness constraints* on statements made both by trusted intermediaries (certifiers) and by principals that may be authenticated. These statements represent assertions whose authenticity can be protected using a variety of mechanisms ranging from public- or shared-key to physical protection. Freshness constraints, which restrict the useful age of statements, can come from initial authentication assumptions and can also be derived from authentic statements which may themselves be subject to freshness constraints.

An important requirement of revocation in large distributed systems is the *fail-safe* property. This means that revocation is resilient to unreliable communication. Revocation mechanisms not satisfying this property can be impeded by active attacks in which the adversary prevents the reception of revocation statements. Apparent countermeasures to these attacks may not be adequate. For example, consider the technique of cascaded delegation where a delegation certificate is created as a delegation is passed to each new system [8]. To terminate a delegation, a “tear down” order is passed down the chain. (This is analogous to the notification mechanism discussed in Section 4.3). However, due to unreliable communication or a compromise of an intermediate system, the order may not fully propagate. Therefore, it was proposed that each intermediate delegate periodically authenticate delegates. However, periodic authentication of predecessor delegates can be vulnerable to attacks where the adversary steps down the chain blocking revocation statements until the particular link times out. The result is an additive effect on delaying revocation. Alter-

[1] Time-stamped based authentication is vulnerable to clock synchronization failures and to inaccuracies in an authenticating entity’s knowledge of certifier information caused by a wide time window.

natively, each node could authenticate every other node at the cost of $n^2$ messages where $n$ is the number of nodes. The optimal design for balancing performance and security depends on the protection of each system and communication between them.

Performance optimizations and increased assurance can be realized if policies for querying entities could be fine-grained and the reasoning behind these policies could be formally analyzed. Recent-secure authentication is necessary both during initial authentication as well as during a session since revocation can occur at any time.

Principals can be used as the basis for specifying freshness constraints and can be typed according to freshness classes. The freshness constraint associated with a principal type can depend on an identification authority's certificate issue policy since this policy specifies security-relevant information such as the randomness of keys, software and hardware protection for key management, identification, and authentication requirements for certificate registration.

Communication latency is an inherent property of distributed systems. Consequently, authenticating entities can not have perfect knowledge of authentication and authorization information and so failure can occur. The problem is compounded as systems grow in size. Additional certifiers represents more distributed knowledge. Our method makes precise what degree of protection is desired and obtained.

By adjusting the recent-secure constraints the delay for revocation can be arbitrarily bounded.This is important since zero delays can not be achieved in large distribute systems due to communication latency.

Given that obtaining consistent knowledge of authentication data may be difficult and cost prohibitive in practice, it is necessary to quantify levels of protection that can be obtained and to reason whether they have been obtained. The practical significance of the concept of recent-secure authentication is that it enables distributed authenticating entities, on a per-transaction basis, to trade-off authentication costs against the levels protection.

Quantifiable authentication assurances are difficult to provide if information about intermediate system is incomplete. In spite of this many systems operate with incomplete information. This requires the risk of operating such systems to be periodically reassessed. For example entire industries have been dependent on reassessing shifting risks. We suggest that recent-secure authentication policies are an important variable for reassessing shifting risk in large distributed systems. For example, proposals have been made to assign financial liability attributes to certificate authority statements in financial systems based on shifting risk [2]. Our method suggests that recent-secure policies should be associated with statements of authentication liability.

The concept of recent-secure authentication/authorization is related in spirit to other systems concepts. In particular, it can be related to a hybrid optimistic/pessimistic method of concurrency control that allows for the selective application of the most efficient method characterized by the probability of conflict [12]. For example, small freshness intervals correspond to pessimistic methods requiring more expensive mechanisms than those required by larger intervals.

## 3. THEORY

In this section, we present a theory for specifying temporal features of statements made by entities, and we present rules for reasoning about them. A primary purpose is to understand how to attain recent-secure authentication rather than to specify implementation algorithms for maintaining recent-secure channels. Therefore, an analysis instance does not carry additional constraints as extra baggage to the conclusion and the conclusion is only meaningful at the time of verification. Also, the theory does not reason about the liveliness properties of an architecture for attaining these properties. Nor do we describe algorithms for finding trustworthy certifiers [9,27].

Our work is inspired by recent literature on a theory of authentication in distributed systems [1,14]. This theory explains how one can reason about authority to deduce other principals that can speak for an entity. The theory has been useful for explaining general techniques of authentication in distributed systems including how one derives the adequacy of a key used to authenticate an entity. Reasoning about the freshness and temporal validity of authentication statements is not addressed to any work to date. Where possible, we build on existing theory to realize its expressiveness for reasoning about general remote procedure calls, name lookup, groups, program loading, delegation, and access control. However, applications of recent-secure authentication to these techniques is not described herein.

We begin by briefly reviewing the theory of secure channels (we refer the reader to the literature for a more detailed description [1,14]) and discuss assumption about time and ordering. Next, we discuss statements about time followed by some example interpretations. Finally, we present a formal description of the syntax, and axioms. A formal semantics, extension axioms, and theorems derived will be presented in a later paper.

## 3.1 Principals, Statements, Time and Ordering

***PRINCIPALS***

All entities in our system are called *principals*. A distinguished principal is the *authenticating entity* who authenticates a channel. Basic *named principal*s are entities that can not say things directly on a computer. For example, they can be people, groups, or roles. *Channels* or channel principals are principals that can say things directly. An I/O port is an example of a channel that reports the receipt of a message. A key and cryptographic algorithm is an example of a channel that makes cryptographic statements. Cryptographic channels are of primary interest when communication transits untrusted domains and is subject to wiretapping.

***STATEMENTS***

If $K_{Bob}$ and *Bob* are principals than,

$$K_{Bob} \Rightarrow Bob$$

is a statement. The $\Rightarrow$ is a '*speaks for*' relation. Suppose $K_{Bob}$ is a channel and *Bob* is a named principal, then the above statement let's us deduce that the channel $K_{Bob}$, represents *Bob*.

If *IA* is a principal and *s* is a statement then,

$$IA\ \mathtt{says}\ s$$

is also a statement. If "*IA* **says** s", we can proceed as though *IA* is willing to say *s*. It does not necessarily mean that *IA* had actually articulated the statement. For example, we may have derived this from our axioms. Also, a principal could be lying.

***CLOCKS AND ORDERING ASSUMPTIONS***

For acceptable performance, basic channel principals can have clocks that are loosely synchronized to an external time reference and the channel's synchronization accuracy is known to other principals and the authenticating entity. Loosely synchronized clocks are used to expire keys, and to age statements. The accuracy of clock synchronization constrains the granularity for aging statements and expiring keys. Granularity constraints on expiring keys is not a practical problem in situations where sufficiently pessimistic assumptions can be made in assigning key lifetimes [25]. However, granularity is an issue for fail-safe revocation since the practical bound on revocation may need to be on the order of tens of minutes or less. The reliance on clock synchronization for refreshing statements makes recent-secure authentication susceptible to vulnerabilities due to clock failures [4,10,16]. Therefore, the assumption of synchronized clocks must be carefully scrutinized.

We assume statements from each channel principal can be ordered and missing statements can be detected. (In practice this requirement might be carefully relaxed. For example, if statements could be ordered and each statement provides a complete interpretation then interpretation of missing statements may be unnecessary.) Also, the order of statements from different sources can be established using clock synchronization and the external synchronization accuracies can be determined.

## 3.2 Statements about Time

***SAYS WITH A TIME ATTRIBUTE***

A principal may assert a statement and a time attribute. For example,

$$K_{IA}\ \mathtt{says}\ s\ \mathtt{at}\ t.$$

In the above statement, it is not necessarily a fact a principal says *s* at time *t*. As mentioned before, a principal could be lying. However, our axioms capture the notion that if we trust a principal and are able to discern a statement made by it, then we also trust the facts concerning the statement time attribute. Not all "says" statements have time attributes. An example is the interpretation of the X.509 identification certificate [28].

***VALIDITY CONSTRAINTS ON 'SPEAKS FOR' RELATIONS***

Speaks for relations are given time constraints. We specify these constraints using the 'notbefore', and 'notafter' suffixes. For example,

$$K_{Bob} \Rightarrow Bob\ \mathtt{notbefore}\ t_1\ \mathtt{notafter}\ t_2$$

indicates that during the closed interval $[t_1, t_2]$, $K_{Bob} \Rightarrow Bob$.

## 3.3 On Interpretation & Analysis

For analysis, we need to first interpret individual messages (such as public key certificates) as statements in our formalism. The next step is to determine that we have an unambiguous interpretation.[2] If so, we proceed with the analysis using the axioms in this paper.

Below, we informally give examples of interpreting authentication message types (e.g., certificates) and formalizing them as statements.

***IDENTIFICATION CERTIFICATES***

In practice, an identification certificate (sometimes called a certificate) contains identifying information of an entity together with its associated keying material and possibly other information such as an expiration date. The identification certificate is cryptographic protected by using the key of an identification authority. The interpretation of a X.509 and Internet Society's (X.509 compliant) IPRA identification certificate [13] is represented as:

[2] The method for determining a consistent interpretation for a given set of statements is not presented herein.

$$K_{IA}\ \texttt{says}\ (K_{IA/Bob} \Rightarrow IA/Bob\ \texttt{notbefore}\ t_1\ \texttt{notafter}\ t_2)$$

In this statement, $K_{IA}$, which is the identification authority's key, asserts the $\Rightarrow$ relationship between $K_{IA/Bob}$ and *IA/Bob* between the validity interval $[t_1, t_2]$. Since a timestamp is not in the identification certificate, we can annotate the time. Consequently, from the statement alone, the authentication entity can not determine the recentness of the statement.

***TIME-STAMPED IDENTIFICATION CERTIFICATE***

Where the identification authority can be adequately protected, architectures may use identification certificates as the primary basis for determining freshness. Since none of the current standards [2,13,29] specify a timestamp within the identification certificate, we assume the X.509 certificate format with a timestamp and obtain:

$$K_{CA}\ \texttt{says}\ (K_{CA} \Rightarrow Bob\ \texttt{notbefore}\ t_1\ \texttt{notafter}\ t_2)\ \texttt{at}\ t_3.$$

***REVOCATION CERTIFICATE***

A certificate revocation list (CRL) or revocation certificate indicates what certificates have changed in "status". The interpretation of a revocation certificate typically has the format of the time-stamped identification certificate. We summarize interpretations of revocation certificates as follows:

"*current*"*:* The absence of a referenced identification certificate implies that it is current at the time of the revocation certificate. Our interpretation asserts the interpretation of the referenced identification certificate with the time of the revocation certificate.

"*revoked*"*:* indicates the referenced certificate is invalid or is about to be invalid. Our interpretation reasserts the original statement specifying a 'notafter' time corresponding to the revocation date. If the timestamp is present we annotate the time of the statement.

"*extended*"*:* indicates the certificate binding is extended to the new time. Our interpretation reasserts the original statement and the 'notafter' time is set for the new time. If the timestamp is present we annotate the time of the statement.

"*suspended*": indicates the certificate should be temporarily suspended (analogous to the certificate on hold option in [2]). Our interpretation consists of two statements: a statement indicating a 'notafter' time set sooner than the original certificate, and a new statement with a 'notbefore' time set later. If the timestamp is present we annotate the time of the statement.

## 3.4 Syntax and Axioms

We now present the axioms of the paper, most are extensions of [1,14] and others are repeated for completeness. The notation $\vdash s$ means that $s$ is an axiom of the theory or is provable from the axioms. The symbol, '$\supset$' means implies, and '$\Rightarrow$' is the 'speaksfor' relation between principals. The symbol $t_{now}$ represents the time of verification.

***SYNTAX***

Formulas are defined inductively, as follows:

- a countable supply of primitive propositions $p_0, p_1, p_2, \ldots$ are formulas;
- If $s$ and $s'$ are formulas then so are $\neg s$ and $s \wedge s'$;
- If $A$ and $B$ are principals expressions then the following are formulas:

$$A \Rightarrow B\ \texttt{notbefore}\ t_1\ \texttt{notafter}\ t_2\ ;$$
$$A\ \textbf{says}\ s\ \textbf{at}\ t\ ;$$

***STATEMENT AXIOMS***

Statements are inductively defined according to the following axioms.

The axiom for statements are:

(S1) If $s$ is an instance of a propositional-logic tautology then $\vdash s$.

(S2) If $\vdash s$ and $(\vdash s \supset s')$ then $\vdash s'$.

(S3) $\vdash (A\ \texttt{says}\ (s \supset s')\ \texttt{at}\ t) \supset ((A\ \texttt{says}\ s\ \texttt{at}\ t) \supset A\ \texttt{says}\ s'\ \texttt{at}\ t))$.

(S4) If $\vdash s$ then $\vdash A\ \texttt{says}\ s\ \texttt{at}\ t_{now}$, for every principal $A$.

From (S1)-(S4) we obtain the theorem:

(S5) $\vdash A\ \texttt{says}\ (s \wedge s')\ \texttt{at}\ t \equiv (A\ \texttt{says}\ s\ \texttt{at}\ t \wedge A\ \texttt{says}\ s\ \texttt{at}\ t)$

***AXIOMS FOR PRINCIPALS***

(P1) $\vdash$ $\wedge$ is associative, commutative, and idempotent.

(P2) $\vdash$ $|$ is associative.

(P3) $\vdash$ $|$ distributes over $\wedge$ in both arguments.

The time of a statement made by $(A \wedge B)$ is equivalent to both $A$ and $B$ saying it at the same time.

(P4) $\vdash (A \wedge B)\ \texttt{says}\ s\ \texttt{at}\ t \equiv (A\ \texttt{says}\ s\ \texttt{at}\ t) \wedge (B\ \texttt{says}\ s\ \texttt{at}\ t)$

$B$ quoting $A$ at time $t$ has the definition:

(P5) $\vdash (B|A)\ \texttt{says}\ s\ \texttt{at}\ t \equiv B\ \texttt{says}\ (A\ \texttt{says}\ s\ \texttt{at}\ t)\ \texttt{at}\ t$

We introduce a new axiom, (P6), that allows more restrictive $\Rightarrow$ relations to be obtained from less restrictive ones. It is used to normalize suffix constraints prior to applying other rules such as the transitivity of $\Rightarrow$.

(P6) $\vdash (A \Rightarrow B\ \texttt{notbefore}\ t_1\ \ \texttt{notafter}\ t_2) \supset (((t_1 \leq t_3) \wedge (t_4 \leq t_2)) \supset A \Rightarrow B\ \texttt{notbefore}\ t_3\ \ \texttt{notafter}\ t_4)$

In (P7), the $\Rightarrow$ relation allows us to remove a level of indirection. The constraint $t_1 \leq t_3 \leq t_2$, is meaningful if principals are trusted not to lie. For example, revocation authorities in the example in Section 6 are trusted not to lie when specifying the time of revocation certificates.

(P7) $\vdash (A \Rightarrow B$ **notbefore** $t_1$ **notafter** $t_2) \supset (((A$ **says** $s$ **at** $t_3) \wedge (t_1 \leq t_{now}, t_3 \leq t_2)) \supset (B$ **says** $s$ **at** $t_3))$

The hand-off axiom allows principals to derive new facts:

(P8) $\vdash (A$ **says** $(B \Rightarrow A$ **notbefore** $t_1$ **notafter** $t_2)$ **at** $t_3) \supset (B \Rightarrow A$ **notbefore** $t_1$ **notafter** $t_2)$

## 4. RECENT-SECURE AUTHENTICATION AND MECHANISMS

This section we explain how recent-secure authentication, and hence bounds on revocation, are specified as freshness constraints on statements made by trusted entities. A discussion of recent-secure authentication in Kerberos is presented, and we briefly review techniques for revocations and discuss their relative merits.

### 4.1 Policy Constraints of Authenticating Entities

To effect revocation, authenticating entities are assumed to follow certain freshness policies. These policies can come from the authenticating entity's participation in a distributed service. For example, a vender conducting electronic commerce on a public network authenticates customers according to the policies of organizations taking financially liability for the transaction. Authenticating entities can also impose their own policies as necessary.

Authenticating entities are given a set of credentials or statements for channel authentication. They also begin with a set of freshness constraints that embody, in part, the revocation policies. Revocation policies are also embodied in freshness constraints recommended by intermediary certifiers. That is,

***Defn***. *$A \Rightarrow B$ satisfies freshness constraint $\delta_{A \Rightarrow B}$ at time $t_{now}$ iff $A \Rightarrow B$* ***notbefore*** *$t$ where $(t_{now} - \delta_{A \Rightarrow B}) \leq t$* and $\delta_{A \Rightarrow B} \leq t_{now}$.

Freshness constraints can be applied to $\Rightarrow$ relations during interpretation by replacing existing less restrictive "notbefore" qualifiers with the more restrictive freshness constraints. For example, if the freshness constraint $\delta_{CA \Rightarrow CA/Bob}$ = 30 minutes is assumed by the authenticating entity, and we are given $CA \Rightarrow CA/Bob$, then we obtain:

$CA \Rightarrow CA/Bob$ **notbefore** ($t_{now}$ - 30 minutes).

Channels obeying freshness constraints are called *recent-secure channels* and are defined as follows:

***Defn***. *$A \Rightarrow B$ is recent-secure at time $t_{now}$ iff $A \Rightarrow B$ can be deduced from given statements with freshness constraints applied.*

It follows from our definitions that if freshness constrains are associated with each trusted certifier whose statements may be used to establish a channel, then the delay for revocation can be bounded by the least restrictive freshness constraint. Consequently, the choice of freshness constraints can arbitrarily bound the delay for revocation. Certifiers recommend freshness constraints typically by imposing "notafter" times in certificates. (However, in Section 5 we illustrate how freshness policies can be promulgated in certificates.)

### 4.2 Recent-Secure Authentication in Kerberos

Kerberos is designed to enable application servers to impose freshness constraints on users who initially log in using shared keys. To do this, the time of initial log in "authtime", is carried in a ticket that is presented to the application server. Application servers can choose to reject the credentials if the "authtime" is not recent even if the ticket is still within its valid lifetime.[3]

A recent proposal calls for public key extensions for initial authentication [20]. With the exception of public-key authentication during initial log in, all protocols remain the same. The implication of recent-secure authentication is that application servers need to know that their recent-secure authentication requirements are satisfied.

With the addition of public-key authentication to Kerberos, the current "authtime" field may not be sufficient for application servers to determine if initial authentication satisfies their authentication policies. The problem is complicated by the fact that recent-secure authentication policies may vary for each server, and possibly, each particular type of transaction. To make Kerberos "recent-secure friendly", one approach would be to require users to satisfy prescribed recent-secure authentication policies prior to obtaining a ticket. During the course of a session, the application server may require the user to satisfy new policies or simply maintain freshness policies during a long session (e.g., refresh stale certificates). A new ticket can be issued once recent-secure authentication is satisfied.

### 4.3 General Revocation Mechanisms

A revocation service should have the properties of being definite, available and recent, contained, bounded recovery, and adjustable, as noted in the introduction. A

[3] This example was pointed out to me by Clifford Neuman.

number of related techniques have been proposed for effecting revocation in distributed systems. We briefly review these techniques and discuss their relative merits for use in large distributed systems.

### *Certificate Caches with Exception Notifications*

Authenticating entities may cache certificates and notify caches when there is a change. This approach is not well suited to large distributed systems since the notification mechanism is not fail-safe. For example, an adversary could selectively block an exception notification. Also, it does not scale well if the destination caches need to be tracked. However, emergency notifications can augment a fail-safe scheme to possibly shorten revocation delays (provided messages reach their destinations sooner than the time-out periods). A distributed multicast infrastructure could alleviate the load on servers for distribution of notifications.

### *Certificates with Expiration Times*

A common technique for bounding the delay of revocation is placing explicit expiration times within certificates. Provided a certifier has not been compromised, statements using expiration times satisfy the fail-safe property. However, since authentication can depend on trusted intermediaries, an entity might be vulnerable to illegitimate statements made by a compromised certifier. Consequently, neither the certifier nor the authenticating entity can be assured that it or its subordinates have not been cloned due to the compromise of an arbitrary certifier that is trusted by an authenticating entity.

### *On-Line Servers and Quorum Schemes*

On-line servers have been proposed whereby entities issue queries in an authenticated exchange to learn the validity of authentication/authorization information. Use of on-line servers may be justified in architectures where the server is local to the source or destination. However, network failures can significantly impact the availability of such servers for geographically distributed clients.

Replicating trusted servers, to increase the availability, inherently increases the risk of compromising the secret keys held by the server. Secret-sharing techniques can improve availability and security [11], but they do so at the expense of considerable communication costs and increased delay. For example, the effective time of the statement from the quorum might be the earliest statement time in a final round used to make the decision. Due to communication costs and increased delay, geographic distribution of secret-sharing servers, for the purposes of surviving network failures, may not be practical for most applications.

### *(Long-Lived) Certificates and Periodic Revocation Statements*

Revocation methods have been proposed where authorities issue long-term identification certificates and also periodically publish timestamped revocation certificates. Revocation certificates can be distributed to the authenticating entity through untrusted communications.

The scalability of this approach depends on whether servers are replicated. However, replicating the trusted identification authority inherently decreases security. In this case, a compromised server may enable the adversary to issue new identification certificates.

### *Off-line Identification Authority and On-line Revocation Authority*

An approach for increasing the availability and security of a authentication service calls for joint authorities [14]. An off-line identification authority generates long-term certificates and an on-line revocation authority creates countersigned certificates with short lifetimes. The effective lifetime is the minimum lifetime of both certificates.

The joint authority approach benefits from the fact that the compromise of the on-line server does not enable the adversary to issue new identification certificates. As expected, a compromised revocation authority could delay revocation until the authority of the on-line server expires. However, the period of *a compromise may be extended further* if the revocation authority issues revocation certificates with longer lifetimes.

An alternative approach to creating countersigned certificates is to authenticate a channel to an on-line (trusted) database server and to retrieve original certificates. However, authenticated retrieval of certificates alone may be insufficient to provide adequate assurance that the certificate is current. For example, to provide high availability for geographically distributed clients, the revocation service might replicate the database and use optimistic consistency control techniques. These techniques do not guarantee the consistency of stored certificates at the time of retrieval. Consequently, the presence of a certificate in a local replica might represent stale information. Additional delays occur as certificates are exported to trusted subsystems for local retrieval.

Also, the on-line revocation server/database is subject to the scaling limitations inherent to on-line servers (discussed previously).

## 5. FRESHNESS-CONSTRAINED REVOCATION AUTHORITY

In this section, we present an improvement to the technique of a separate identification authority and revocation authority. The technique enables the identification authority to retain control on the specification of revocation policies. Also, information about the freshness of revocation certificates is made explicit. The practical significance of this technique is that it can be used to build a secure revocation service. The approach is a step towards enabling the consolidation of revocation services since less trust is required of the service.

### 5.1 Revocation Service

With the emergence of large distributed directories, we investigate improvements to the joint authority technique for building a highly available and secure revocation service. Our approach uses identification authorities for issuing long-lived identification certificates and revocation authorities issuing *timestamped certificates*. (Timestamped certificates do not require expiration dates.) The identification authority and the revocation authority have only *unidirectional communication* to the network.[4] A highly available mechanisms is needed to enable the posting of updates to the revocation service.

The identification authority specifies freshness constraints for revocation within the identification certificate. That is, in addition to the conventional key and name binding, the revocation authority is also designated. However, the identification authority specifies the restriction that revocation certificates (obtained from the revocation authority) meet the designated freshness constraint. The significance of specifying freshness constraints in the identification certificate (yet delegating revocation authority) and issuing timestamped revocation certificates are as follows:

- *Less Trust.* The revocation authority need not be trusted to assign correct lifetimes to short-term revocation certificates. (Otherwise, even after the compromised revocation authority is discovered, these certificates might still be used.)
- *Secure.* Clients need not interact with trusted on-line servers containing keys. That's because high availability relies on data replication using untrusted servers (instead of replicating trusted processes). It is also secure because both identification and revocation authorities have unidirectional communication to the network.
- *Scalable.* Using timestamped revocation certificates instead of certificates with explicit expiration times scales better since the same timestamped certificate can by used by any authenticating entity and can be used for multiple freshness policies.
- *Highly Available*. Availability is possible assuming the use of directories.

### 5.2 Formal Description

We now formally describe the statements of the revocation service. Derivation of channels from these statements is described in the example in Section 6.

Entities wishing to establish a channel for *B* require a secure channel with the identification authority, *IA*, and must trust that *IA* has the authority to issue a certificate to *B*:

(1) $K_{IA} \Rightarrow IA$

(2) $IA \Rightarrow B$

The off-line identification authority issues the following certificate:

(3) $K_{IA}$ **`says`** $((K_B \Rightarrow B$ **`notbefore`** $t_1$ **`notafter`** $t_2) \wedge (RA|K_B \Rightarrow B$ **`notbefore`** $(t_{now} - \delta_{IA \Rightarrow B})))$

This statement has several important features. First, the expiration time of the certificate for *B* is long lived. Second, *IA* imposes its recent-secure policy, $\delta_{IA \Rightarrow B}$, on *B* using the constraint, $RA|K_B \Rightarrow B$ **`notbefore`** $(t_{now} - \delta_{IA \Rightarrow B})$. This means that for $K_B$ to be valid, claims made by $RA|K_B$ must be interpreted according to *IA*'s freshness policy stated in this certificate.

The revocation authority, *RA*, states the identification certificate is current.

(4) $R|K_B$ **`says`** $(K_B \Rightarrow RA|K_B$ **`notbefore`** $t_3$ **`notafter`** $((t_3 + \delta_{IA \Rightarrow B}))$ **at** $t_3$

The particular value of $\delta_{IA \Rightarrow B}$ is not interpreted from the message corresponding to statement (4). Instead it is obtained from the original certificate statement (3).

More elaborate forms of joint authority might be useful. For example, using disjunction we can specify a number of revocation authorities.

## 6. AN EXAMPLE

In this section, we apply our method by specifying and analyzing a hypothetical authentication instance of an experimental architecture illustrated in Figure 1. Identification and revocation certificates are indicated by directed arrows from the certificate authority to the entity designated in the certificate. The delegation certificate is indicated by the dotted arrow. Also, illustrated is a replicated directory organized for the propagation of

[4] It may be desirable to complement this approach (possibly when revocation certificates must be extremely fresh and propagation delays in *any* conceivable directory is too slow). One approach might be to use an on-line secret-sharing scheme with low latency and highly reliable communication between the secret-sharing servers. When authenticating entities are able to access the central service, they could obtain fresh revocation certificates.

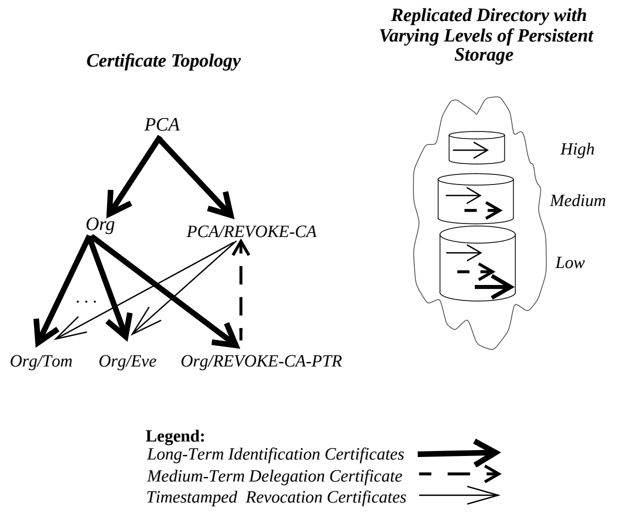
Certificate Topology
Replicated Directory with Varying Levels of Persistent Storage
PCA
Org
PCA/REVOKE-CA
Org/Tom
Org/Eve
Org/REVOKE-CA-PTR
High
Medium
Low
Legend:
Long-Term Identification Certificates
Medium-Term Delegation Certificate
Timestamped Revocation Certificates

**Figure 1. Experimental Revocation Architecture**

certificates according to the recent-secure delay bounds of authenticating entities. We need not assume the directory is trusted. This example illustrates:

- recent-secure authentication using freshness constraints imposed by the authenticating entity and constraints recommended by intermediary certifiers.
- a technique for delegating revocation authority without requiring the revocation agent be trusted to specify revocation policies.
- how indirection in long-term certificates provides flexibility for changing revocation agents.
- how the addition of timestamps to X.509 like identification certificates may improve the performance for satisfying freshness constraints in specialized architectures.

### *Channel Authentication Goal*

In our scenario, the authenticating entity's goal is to authenticate that public key, $K_{Org/Eve}$, can be used to establish a secure channel with *Org/Eve.* That is,

(1) $K_{Org/Eve} \Rightarrow Org/Eve.$

### *Recent-Secure Policy Constraints*

Suppose the authenticating entity is required to obey the following freshness constraints (perhaps, due to the particular value of the transaction to be authenticated):

(2) $\delta_{PCA \Rightarrow Org}$ = 31 days

(3) $\delta_{Org \Rightarrow Org/Eve}$ = 30 minutes

Also, the authenticating entity is to obey constraints recommended by all trusted intermediaries.

### *Time of Authentication*

The time of authentication is:

(4) $t_{now}$ = *14:00 6/15/95*

### *Assumptions*

Let's suppose, for the particular type of transaction, the authenticating entity can only trust entities within the a particular hierarchial *security domain* indicated by a top level policy certification authority called *PCA.* (A security domain indicates that principals have compatible security policies.) Also, the authenticating entity trusts the *PCA* to certify the parent organization of the principal we wish to authenticate. That is,

(5) $PCA \Rightarrow Org$

Also, (perhaps due to the desired security assurance and liability arrangements), let's assume the authenticating entity is required to initiate the authentication chain starting from the top level *PCA* certificate. Initially, we assume the authenticating entity knows the top level key of *PCA:*

(6) $K_{PCA} \Rightarrow PCA$ **`notafter`** *00:00 8/01/95*

For brevity, assume certifying authorities (subordinate to the *PCA*) may only assign certificates using hierarchial names subordinate to their own names. That is,

(7) $PCA \Rightarrow PCA/REVOKE\text{-}CA$

(8) $Org \Rightarrow Org/Eve$

(9) $Org \Rightarrow Org/REVOKE\text{-}CA\text{-}PTR$

However, the policy within the security domain allows specially designated delegation certificates to indicate other trusted entities within the same security domain. An example is the medium-term delegation certificate indicated by a dotted arrow in Figure 1.

### *Given Statements and Discussion*

Suppose the authenticating entity is presented with a long term X.509 certificate for *Org* interpreted as follows:

(10) $K_{PCA}$ **`says`** ($K_{Org} \Rightarrow Org$ **`notafter`** *00:00 1/1/96*)

Also, the most recent monthly X.509 certificate revocation list is presented to the authenticating entity:

(11) $K_{PCA}$ **`says`** ($K_{Org} \Rightarrow Org$ **`notafter`** *00:00 1/1/96*) **at** *00:00 6/1/95*

Recall, the original CRL message need not repeat the field of the original certificate message.

Next, we explain a special certificate for *Org/Eve*. The certificate specifies the key for *Org/Eve*, i.e. $K_{Org/Eve}$, and also designates *Org/REVOKE-CA-PTR* as an authority for revocation. For the particular type of transaction of interest, *Org* assumes liability only if a 30 minute recent revocation certificate statement has been obtained.

(12) $K_{Org}$ **`says`** (($K_{Org/Eve} \Rightarrow Org/Eve$ **`notafter`** *00:00 4/1/96*) $\wedge$ ($Org/REVOKE\text{-}CA\text{-}PTR | K_{Org/Eve} \Rightarrow Org/Eve$ **`notbefore`** ($t_{now}$ - 30 minutes))

The authenticating entity is also presented with the most recent weekly certificate which specifies *Org/REVOKE-CA-PTR* as the entity *Org* trusts to serve as a revocation agent. In this example, *Org/REVOKE-CA-PTR* does not actually exist. It serves as delegation to a particular revocation agent. This techniques enables flexibility in changing the revocation agent without having to reissue the long-term certificate. *Org* satisfies *PCA*'s requirement that the designated entity be within the security domain. Of course, the agent may serve multiple security domains provided it satisfies the policy for each domain. The "notafter" suffix above constraints the exposure the identification authority has to future attacks on the revocation service. The effective bound, however, may extend to the least restrictive freshness constraint in any one of the identification certificates.

(13) $K_{Org}$ **says** (*PCA/REVOKE-CA* ⇒*Org/REVOKE-CA-PTR* **notbefore** *00:00 7/13/95* **notafter** *00:00 7/20/95*)

We could have specified a longer certificate and required a different revocation authority to assert it's status, however, each level of indirection comes at an expense.

Also, we are given the long term identification certificate for a central revocation service:

(14) $K_{PCA}$ **says** ($K_{PCA/REVOKE-CA}$ ⇒*PCA/REVOKE-CA* **notafter** *00:00 1/1/96*).

This revocation agent certifies that *Org/Eve*'s certificate was current at *13:40 7/15/95*:

(15) $K_{PCA/REVOKE-CA}|K_{Org/Eve}$ **says** ($K_{Org/Eve}$⇒*PCA/REVOKE-CA*|$K_{Org/Eve}$ **notbefore** *13:40 7/15/95* **notafter** (*13:40 7/15/95* + $\delta_{Org\Rightarrow Org/Eve}$) **at** *13:40 7/15/95*.

Observe that the timestamp of the certificate message is interpreted as a 'notafter' variable in the above. In our statement the value of $\delta_{Org\Rightarrow Org/Eve}$, is not made explicit but is interpreted from the original certificate statement, (12), issued by *Org*.

The real certificate could contain one or more original certificates and a timestamp applied by *PCA/REVOKE-CA*. Alternatively, a condensed version might contain only a, perhaps shorter, dated list of revoked certificates. Note that the agent can be made secure since real-time interaction with a trusted revocation authority is unnecessary for our purposes.

The primary processing function required by *PCA/REVOKE-CA* is the off-line function of dating and signing statements. Distribution of this information is made secure using unidirectional links to the directory. To prevent denial of service, these links must be highly reliable. Also, multiple revocation agents might be designated.

***Analysis***

We analyze the certificate path starting at the node, *PCA*, trusted by the authenticating entity. The lack of a timestamp in the X.509 certificate prevents us from determining the freshness from the certificate alone, (10); the CRL statement, (11), is needed to determine freshness. Using (5), (6), (10), (P6), (P7), (P8), and transitivity of ⇒ we obtain:

(16) $K_{Org}$⇒*Org* **notafter** *00:00 1/1/96*

The above illustrates why the analysis is only good for the time of verification. That is, though the validity of statement (6) is until *00:00 8/01/95*, the conclusion is a fact that (might wrongfully) be interpreted as being valid until *00:00 1/1/96*.

Using (6), (7), (14), (P6), (P7), (P8), and transitivity of ⇒ we obtain:

(17) $K_{PCA/REVOKE-CA}$⇒*PCA/REVOKE-CA* **notafter** *00:00 1/1/96*

Using statements (8), (12), (16), (P7) and (P8) followed by normalizing the suffix constraint using (P6), we obtain:

(18) ($K_{Org/Eve}$⇒*Org/Eve* **notbefore** ($t_{now}$ - 30 minutes) **notafter** *00:00 4/1/96*) ∧ (*Org/REVOKE-CA-PTR*|$K_{Org/Eve}$ ⇒*Org/Eve* **notbefore** ($t_{now}$ - 30 minutes) **notafter** *00:00 4/1/96*)

Using statements (9), (13), (16), (P7) and (P8), we obtain:

(19) *PCA/REVOKE-CA*⇒*Org/REVOKE-CA-PTR* **notbefore** *00:00 7/13/95* **notafter** *00:00 7/20/95*

Using (P6) to normalize the suffix times on (18), and (19), then applying the transitivity of ⇒, we obtain:

(20) ($K_{Org/Eve}$⇒*Org/Eve* **notbefore** *13:30 7/15/95* **notafter** *00:00 7/20/95*) ∧ (*PCA/REVOKE-CA*|$K_{Org/Eve}$ ⇒*Org/Eve* **notbefore** *13:30 7/15/95* **notafter** *00:00 7/20/95*)

Using statements (15), (17), (P7) and (P8), we obtain:

(21) $K_{Org/Eve}$⇒*PCA/REVOKE-CA* | $K_{Org/Eve}$ **notbefore** *13:40 7/15/95* **notafter** (*13:40 7/15/95* + $\delta_{Org\Rightarrow Org/Eve}$)

We first normalize statements (21) and (20) using (P6) then apply the transitive rule for ⇒, and finally, combine like terms to obtain:

(22) $K_{Org/Eve}$⇒*Org/Eve* **notbefore** *13:40 7/15/95* **notafter** *14:10 7/15/95*

Since, *13:40 7/15/95* ≤ ($t_{now}$ = *14:00 6/15/95*) ≤ *14:10 7/15/95*, we conclude:

(23) $K_{Org/Eve}$⇒*Org/Eve*

## 7. CONCLUSION

We presented a method for fine-grained specification of policies for revocation with bounded delay. In our method, revocation policies are embodied in freshness constraints imposed on channel authentication. Channel authentication according to these policies is called recent-secure authentication. We presented a method for reasoning about recent-secure authentication and introduced a technique for promulgating revocation policies in authenticated statements. We explained how this technique enables the design of secure and highly available systems for revocation.

The practical significance of this work is that it enables authenticating entities, on a per-transaction basis, to tune authentication costs at the expense of protection. This is because it makes precise what degree of protection is desired and whether it is obtained. Formal recent-secure policies provide a basis for both optimizing the distribution of credentials (e.g., public key identification and revocation certificates) and for designing authentication architectures that are resilient to network partitioning or disconnected operations.

This work is being extended to include a ‘receivednotafter’ suffix to signal that the integrity of statements received after this time have an unacceptably high threshold of being compromised. (For a description of analysis and design methods for message integrity thresholds in cryptographic protocols, the reader is referred to the literature [24,25].) Message contents cached prior to integrity time-outs might still be usable. Such techniques enable cached certificates to be used past these periods and enable computational efficiencies due to choices in key sizes. Of course, such techniques require careful analysis.

## Acknowledgments

I would like to thank Mathew Blaze, Mathew Franklin, Virgil Gligor, Li Gong, Jack Lacy, Thomas London, Clifford Neuman, Frank Sudia, and Rebecca Wright for discussions and comments on earlier drafts of this paper. I thank Steve Kent for discussions on the semantics of time intervals in certificates and I particularly wish to thank Mike Reiter for early discussions on this topic.

## 8. REFERENCES

1. Abadi, M., Burrows, M., Lampson, B., and Plotkin, G. A calculus for access control in distributed systems. *ACM Trans. Program. Lang. Syst.*, 15(4), Sep. 1993, pp. 706-734.
2. Accredited Standards Committee X9, Public Key Cryptography Using Irreversible Algorithms for the Financial Services Industry: Part 3: Certificate Management for DSA, Draft, American National Standard X9.30-199X, American Bankers Association, Nov. 19, 1994.
3. Berkowits, S., et al., Public Key Infrastructure Study, National Institute of Standards and Technology, MITRE Technical Report, April, 1994.
4. Denning, D., and Sacco, G., Timestamps in Key Distribution Protocols. *Com. of the ACM*, Vol.24, No.8, Aug. 1981, pp.533-536.
5. Diffie, W. and Hellman, M. New directions in cryptography. *IEEE Trans. Inf. Theor.* IT-22, 6 (Nov. 1976), pp. 644-654.
6. Federal Information Processing Standard 186, Digital Signature Standard (DSS), May 1994, FIPS PUB 186, U.S. Dept. of Commerce National Institute of Standrds and Technology, Gaithersburg, MD., Digital Signature Standard.
7. Gasser, M., Goldstein, A., Kaufman, C., and Lampson, B. The Digital Distributed System Security Architecture. Proc. *12th National Computer Security Conference*, National Institute of Standards and Technology, Baltimore, MD., 1989.
8. Gasser, M., and McDermott, E. An architecture for practical delegation in a distributed system. Proc. *IEEE Symp. on Security and Privacy*, Oakland, 1990, pp. 20-30.
9. Gligor, V., Luan, S., and Pato, J. On inter-realm authentication in large distributed systems. Proc. *1992 IEEE Symp. on Research in Security and Privacy*, May 1992.
10. Gong, L. A security risk of depending on synchronized clocks. *ACM Operating Syst. Rev.*, vol. 26, no. 1, pp. 49-53, Jan. 1992.
11. Gong, L. Increasing Availability and Security of an Authentication Service. *IEEE J. on Selected Areas in Commun.*, Vol. 11., No. 5, June 1993, pp. 657-662.
12. Herlihy, M., and Wehl, W. Hybrid concurrency control for abstract data types. In *Proc.7th ACM-SIGMOD-SIGACT Symp. on Principles of Database Systems (PODS)* (Austin, Tex., Mar. 21-23, 1988), pp. 201-210.
13. Kent, S., Privacy Enhancement for Internet Electronic Mail: Part II: Certificate-Based Key Managements, RFC 1422, IAB IRTF PSRG, IETF PEM WG, Feb 1993.
14. Lampson, B., Abadi, M., Burrows, M., and Wobber, E. Authentication in distributed systems: Theory and practice. *ACM Trans. on Comp. Systs.*, 10(4): 265-310, Nov. 1992.
15. Lamport, L. Time, clocks and ordering of events in a distributed system. *Commun. Ass. Comput. Mach.*, Vol. 21, pp. 558-565, July 1978.
16. Liskov, B., Practical Uses of Synchronized Clocks in Distributed Systems. *Proc. 10th Annual ACM Symposium on Principles of Distributed Computing*, Montreal, Aug., 1991, pp.1-9.
17. National Bureau of Standards. Data Encryption Standard. FIPS Pub. 46, Jan. 1977.
18. Neuman, B. Clifford, and Ts’o, Theodore. Kerberos: An Authentication Service for Computer Networks, *IEEE Communications*, Vol. 32. No. 9, Sept., 1994.
19. Neuman, B. Clifford., Proxy-Based Authorization and Accounting for Distributed Systems. Proc. 13th Intern. Conf. on Distr. Comp. Syst., May, 1993, pp. 283-291.
20. Neuman, C., Tung, B., and Wray, J. Public Key Cryptography for Initial Authentication in Kerberos. Internet Draft, Mar. 95.
21. Open Software Foundation. OSF Distributed Computing Environment Administration Guide -- Core Components Revision 1.0, Update 1.0.2, March 1993.
22. Rivest, R., Shamir, A., and Adleman, L. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM* 21, 2 (Feb. 1978), pp. 120-126.
23. Sollins, K. Cascaded authentication. *Proc. 1988 IEEE Symp. on Research in Security and Privacy*, Apr. 1988, pp. 156-163.
24. Stubblebine, S., and Gligor, V. On Message Integrity in Cryptographic Protocols. *Proc. IEEE Symposium on Security and Privacy*, Oakland, 1992.
25. Stubblebine, S., and Gligor, V. Protocol Design for Integrity Protection. *Proc. IEEE Symposium on Security and Privacy*, Oakland, 1993, pp. 41-53.
26. Tardo, J., and Alagappan, K. SPX: Global authentication using public key certificates. Proc. *14th National Computer Security Conferenc*e, NIST/NCSC, Baltimore, 1991.

27. Yahalom, R., Klein, B., Beth, T., Trust-based Navigation in Distributed Systems, *Computing Systems*, Vol. 7, No. 1, Winter 1994, pp. 45-73.
28. International Telegraph and Telephone Consultative Committee (CCITT), X.500, The Directory: Overview of Concepts, Models, and Services, Recommendation X.500, 1988.
29. International Telegraph and Telephone Consultative Committee (CCITT), X.509, The Directory - Authentication Framework, Nov. 1992.