Commercialization of Digital Signatures

Frank Sudia
Bankers Trust

Rich Ankney
Fischer International

February 19, 1994

1 Introduction

Public-key cryptography is a modern computer security technology that can support the creation of paperless electronic document systems, providing that the user's digital signature on a document can be given sufficient practical and legal meaning.

Such "document architectures" will encompass not only trading partners operating under standard bilateral contracts, but also global multilateral systems in which any entity can (in theory) correspond with any other in a legally provable manner, assuming that proper security controls are observed throughout.

These systems will have enormous commercial significance, since in many cases, cost reductions on the order of 10-to-1 could be realized over current paper procedures [5]. This improvement is sufficiently dramatic that organizations would basically be forced to use them, for economic and competitive reasons, once their practicality had been demonstrated.

Thus far, however, major corporations and banks have declined to invest in these technologies, due to lack of well-defined risk models and auditing standards, and uncertainties regarding legality and liability issues.

No one disputes that paper is a bothersome anachronism in the electronic world, or that verifying pen-and-ink signatures is costly and error-prone. But at least with paper, one retains the basic "contextual controls" of document preparation and physical delivery. On a digitally signed electronic document, on the other hand, there is nothing but the encoded signature. All time, place and manner controls are absent, so nothing distinguishes a valid user signature from one produced by anyone else who somehow obtained their smart card and PIN. It would not take too many multi-million (or multi-billion) dollar losses to erase all the savings produced by this "newfangled" office-automation technology.

Digital signatures will see early use in consumer "electronic coin purse" applications, where exposure is low, and might also be used for wholesale financial transfers, where extremely tight security procedures are already the norm. However, these uses would have little general commercial impact.

Serious investments to commercialize digital signatures will occur only after leading national auditing and legal experts have ruled that these systems contain adequate security controls to warrant reliance in mainstream intra- and inter-corporate business transactions, typically in the $10,000 to $10 million range.

To achieve this goal, we must formulate security controls that reduce the risks to participants in digital signature document systems to the absolute lowest level technically achievable.

2 Scope and Organization

This paper discusses strategies for reducing the risks associated with digital signature systems, with special emphasis on issues related to end-user smart cards. Much of this builds on the use of *public key certificates* [1] and *attribute certificates* [2]. These structures are being included in the forthcoming ANSI standard X9.30 Part 3 [2] for use by systems developers.

Our recommendations are framed in light of the CCITT X.500 Directory standard (1988). While an X.500 style directory would be useful for managing user's certificates, most of these techniques do not require one.

The paper is organized as follows:

The first section present an overview of the relevant technologies, including digital signature mechanisms and other cryptographic techniques, as well as the X.500 Directory structures and concepts used in current standards documents. These techniques and structures are used in the following sections to specify the structure of an electronic document.

The next sections discuss the use of certificates to hold *security policy* and *authorization* information, respectively, and are followed by some extensions to the model to accommodate other current business practices. Since most of these security policies must be enforced by the verifier of a signature (i.e., by their document software), it is important that these concepts be widely understood and consistently implemented.

The final section discusses requirements on smart cards used in the system.

3 Technology Overview

3.1 Public Key Cryptography

In *symmetric* (conventional) cryptography, as shown in Figure 1(a), the sender and recipient share a secret key. This key is used by the originator to encrypt a message and by the recipient to decrypt a message. It may also be used to authenticate a message by computing some function such as a Message Authentication Code (MAC) over the message, using the key; the recipient can be assured of the identity of the originator since only the originator and the recipient know the secret key used to compute the MAC. DES is an example of a symmetric cryptosystem.

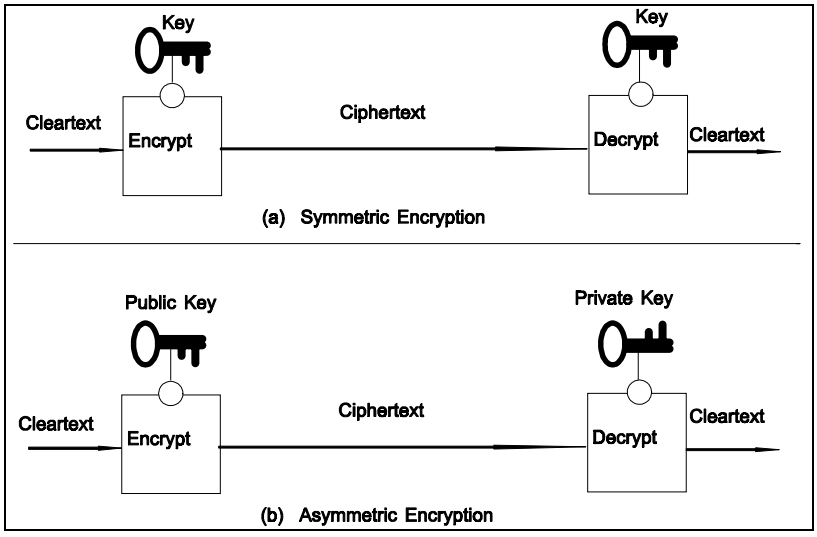

Figure 1. Symmetric and Asymmetric Encryption

In *asymmetric* (public key) cryptography, shown in Figure 1(b), different keys are used to encrypt and decrypt a message. Each user is associated with a pair of keys. To provide confidentiality, one key (the *public key*) is publicly known and is used to encrypt messages destined for that user, and the other (*private*) *key* is known only to the user and is used to decrypt incoming messages. Authentication can be provided using a public key system, too, using the concept of *digital signatures* described below. RSA [1] is the most well-known asymmetric algorithm. Since the public key need not (indeed cannot) be kept secret, it is no longer necessary to secretly convey a shared encryption key between communicating parties prior to exchanging confidential traffic or authenticating messages. Figure 1 illustrates the use of symmetric and asymmetric algorithms for encryption.

Some asymmetric algorithms (e.g., RSA) can also provide authentication and non-repudiation when used as follows: To sign data, the user encrypts it under his private key. To validate the data, the recipient decrypts it with the originator's public key. If the message is successfully decrypted, it must have been encrypted by the originator, who is the only entity that knows the corresponding private key.

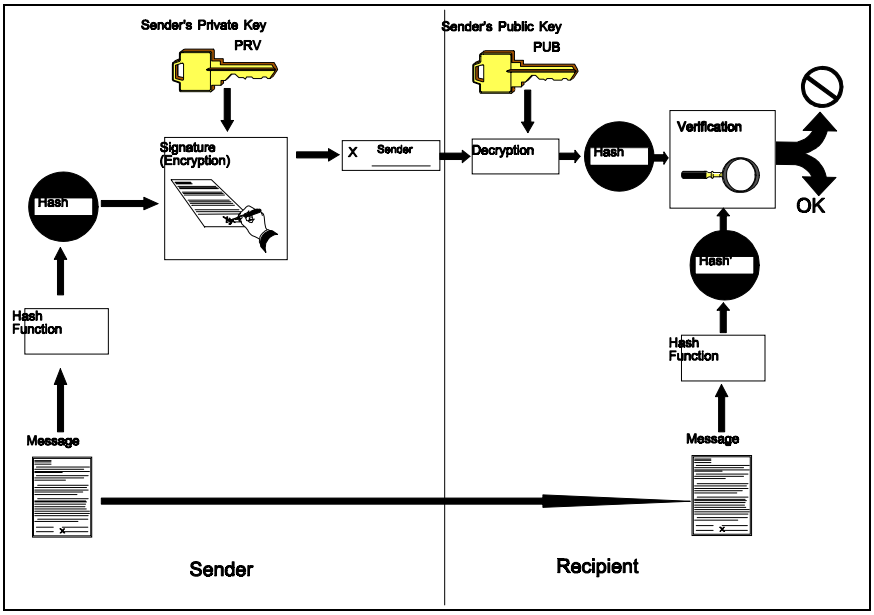

Figure 2. Digital Signatures

A *digital signature* is a piece of data appended to a data unit which allows the recipient to prove the origin of the data unit and to protect against forgery. Digital signatures are formed using asymmetric encryption algorithms as described above. To sign a message, it is first digested (hashed) into a single block using a *one-way function*. A one-way function has the property that, given the digest, it is computationally infeasible to construct any message that hashes to that value, or to find two messages that hash to the same digest. The digest is encrypted with the user's private key, and the result is appended to the message as its signature. Separating the signature from the message reduces the amount of data to be encrypted to a single block. This is important since public key algorithms are generally substantially slower than conventional algorithms. The signature process also introduces redundancy into the message. Redundancy allows the recipient to detect unauthorized changes to the message. Most messages already contain sufficient redundancy to detect such a forgery (e.g., English text, timestamps, etc.). The signature process adds additional redundancy, since the message must also hash to the specified digest. The signature process is illustrated in Figure 2.

A digital signature provides the following security services:

- *Integrity*, since any modification of the data being signed will result in a different digest, and thus a different signature;
- *Origin authentication*, since only the holder of the private key corresponding to the public key used for validation could have signed the message; and
- *Non repudiation*, i.e. irrevocable proof to a third party that only the signer could have created the signature. A traditional secret key authenticator (e.g. the X9.9 MAC) does not provide this service, since either of the two parties can create the authenticator using their shared key.

3.2 Public Key Certificates

For a user to identify another user by his possession of a private key, he must obtain the other user's public key from a source he trusts. A framework for the use of *public key certificates* was defined in CCITT Recommendation X.509 [1]. These basic certificates bind a user's name to a public key, and are signed by a trusted issuer called a *Certification Authority* (CA). Besides the user's name and public key, the certificate contains the issuing CA's name, a serial number, and a validity period.

Although X.509 does not impose any particular structure on the CAs, many implementations find it reasonable to impose a hierarchical structure in which each CA (in general) certifies only entities which are subordinate to it. Hence, we can construct a hierarchy of CAs, where the higher level CAs (perhaps banks) sign the certificates of the CAs beneath them (e.g., companies), etc. The lowest level of CAs sign user certificates. At the top of this hierarchy are a relatively few CAs (perhaps one per country) which may "cross-certify" each other's public keys. Such a hierarchy is shown in Figure 4.

Various security architectures define mechanisms to construct a *certification path* through the hierarchy to obtain a given user's certificate and all CA certificates necessary to validate it. These architectures share the common characteristic that a user need only trust one other public key in order to obtain and validate any other certificate. The trusted key may be that of the top-level CA (in a centralized trust model), or the local CA that issued the user's certificate (in a decentralized model).

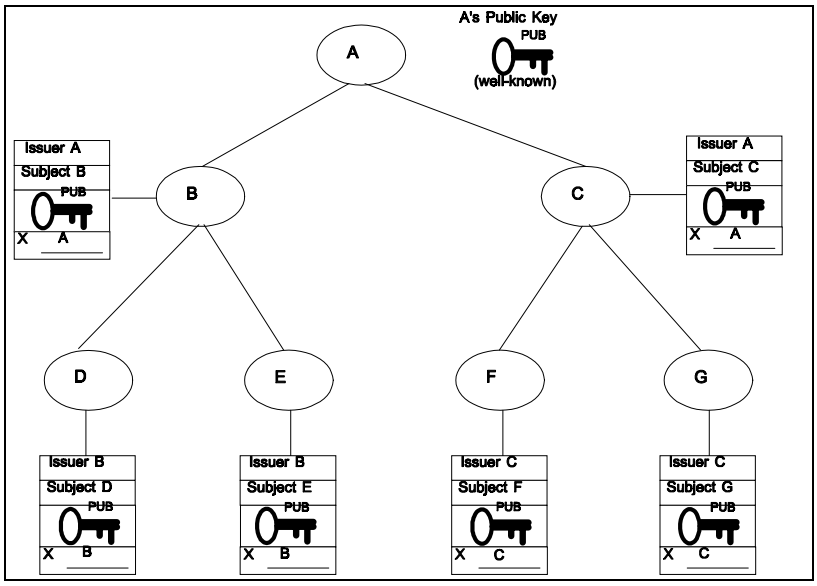

Figure 3. CA Hierarchy

Certificates contain an expiration date. If it is necessary to cancel a certificate prior to its expiration date (e.g., if the name association becomes invalid or the corresponding private key is lost or compromised), the certificate may be added to the CA's *certificate revocation list* (CRL) or "hot list." This list is signed by the CA and widely distributed (e.g., as part of the CA's directory entry). The certificate remains on the hot list until its expiration date.

3.3 Directory Structure

The X.500 Directory structure is hierarchical; the resulting distributed database comprises the *Directory Information Tree* (DIT), as shown in Figure 5. Each entry is of a specific *object class*, and consists of a set of properties called *attributes*. An attribute consists of a type and one or more values. Thus, in an entry of class **organization**, one attribute is the **organizationName**; in an entry of class **organizationalPerson**, attributes might include **title** and **telephoneNumber**.

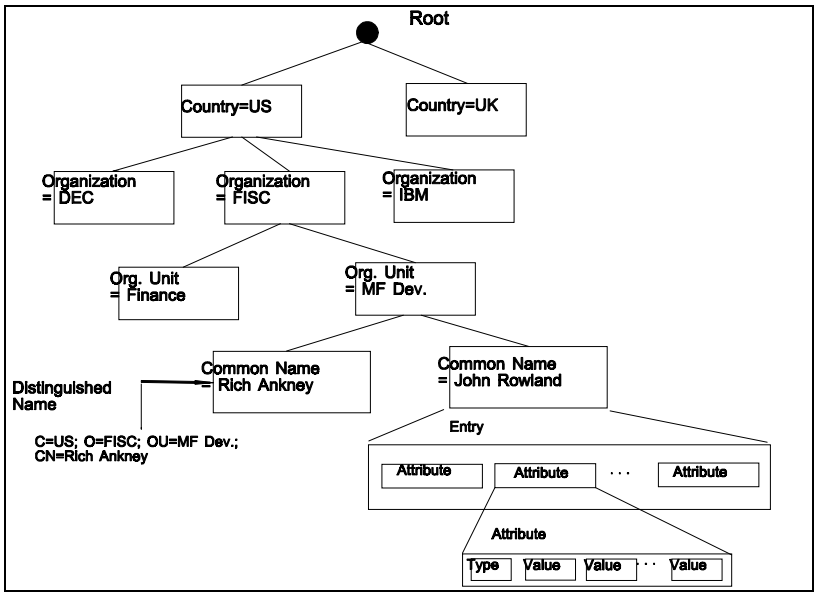

Figure 4. Directory Structure

Each entry has one or more special attribute values used to construct the object's name; this is the *relative distinguished name* (RDN) of the entry. An object's *distinguished name* (DN) uniquely identifies it in the global DIT, and is created by concatenating the relative distinguished names of all entries from the DIT root to the entry.

Several of the attributes defined in X.500 may be usefully included in the user's attribute certificate. For example:

- the *object class* can be used to distinguish between entities (e.g. users and roles) whose distinguished names are of the same form; and

- the *title* may be used in making authorization decisions.

In addition to the use of the DIT to group entities along organizational lines, X.500 defines several object classes which can be used to construct arbitrary groups of entities. These object classes include the *organizational role*, whose *role occupant* attribute lists the names of the users who occupy the role, and the *group of names*, whose *member* attribute lists the names of group members.

3.4 Attribute Certificates

Certain information concerning an entity or CA needs to be made available in a trusted manner. In a secure X.500 Directory, this information would be retrieved via standard Directory operations, and the result would be signed by the Directory. In the absence of such a secure X.500 implementation,

this information is placed in an attribute certificate, which is signed by a CA in the same manner as the public key certificate. This is a separate structure for the following reasons:

- Proper separation of duties might require that a different CA issue the attribute certificate than issued the public key certificate. A central CA might rarely of itself possess the required security or authority to "sign for" all of a user's authorizations. Having separate CAs generate various types of attribute certificates distributes risks more appropriately.

- The defined attributes may not be required for all domains, networks, or applications. The need for these attributes (and for additional domain-specific attributes) is determined by each domain. (This is discussed further in Section 4.)

The user's basic public key certificate remains X.509 compatible, allowing its use with other applications and allowing use of commercial products for certificate generation.

Attribute certificates would be created on presentation of the proper credentials by the user. For example, the user would obviously present his public key certificate and prove he possesses the corresponding private key, as one form of identification. Attribute certificates are linked to the user's basic public key certificate by referencing its serial number, and are revoked by an identical CRL mechanism.
Many authorization decisions may be based on the user's position in an organization. The organizational structure may be specified as part of a user's name. Names in certificates are specified in terms of the X.500 Directory model, which was discussed previously.

To convey this information in a trusted way, one could define role and group certificates which convey the names of the role occupants or group members, respectively, and which are signed by a CA, thus enabling use of this feature outside the context of an X.500 directory system.

4 Document Structure

This signature verification model requires the ability to attach multiple signatures to a document, as well as the ability to include per-signer information in the signature computation. [8] defines a useful format for this purpose, as shown in Figure 3.

4.1 Multiple Signatures

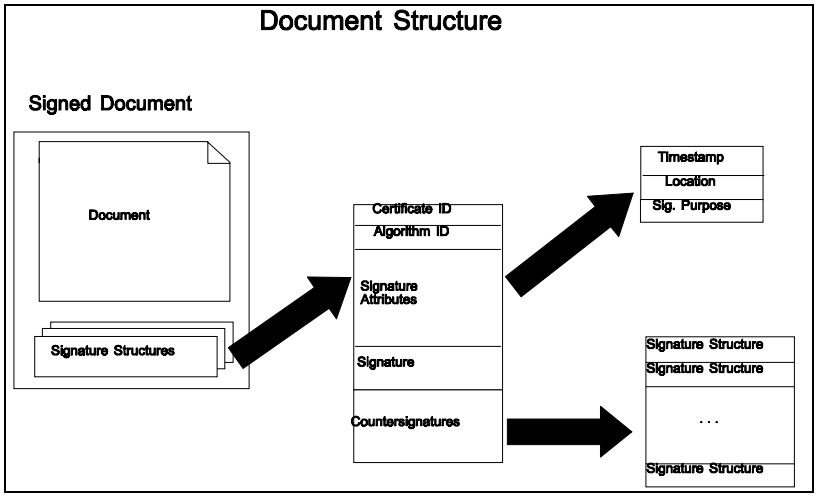

Figure 5. Multiple Signatures

The document structure consists of the actual document, along with a signature structure for each signer of the document. Each signature structure contains an indication of the certificate needed to validate the signature, and a bit string containing the actual signature. Additionally, other information relevant to the particular signer may be included in an individual signature computation. This per-signer information, may be included in the signature computation, as *signature attributes* (see below). A signature structure may also include *countersignatures*. A countersignature is a signature on the signature structure in which it is found, rather than on the document itself. A countersignature thus provides proof of the order in which signatures were applied. Since the countersignature is itself a signature structure, it may itself contain countersignatures; this allows construction of arbitrarily long chains of countersignatures.

4.2 Signature Attributes

Useful signature attributes might include timestamps, location information, and signature purpose [9].

We can distinguish authorizing signatures, which must meet the restrictions specified in the signer's certificate, from other cosignatures by including an indication of the signature purpose in the data being signed, by including the signature purpose as a signature attribute. This *signature-purpose* attribute might have the values:

- *authorization:* authorization signature appropriate to the document,
- *cosignature:* authorization cosignature appropriate to the document; cosigner's certificate has sufficient authority to authorize the document, and
- *witness:* witness cosignature; cosigner's does not have sufficient authority to authorize the document.

Additionally, one might use a signature structure as a signed receipt on a transaction, in which case additional signature purposes for receipt confirmation might be defined. Application-specific purposes (e.g., medical record release) might also be defined.

A user may indicate the role s/he is acting in by including the role in the signature computation, as a (per-signer) signature attribute. The asserted role may be matched against a role certificate (or the user's attribute certificate) during verification. This approach eliminates the need for multiple users to share a public key certificate (and corresponding private key) assigned to the role.

5 General Principles

The following general principles and philosophies are reflected in the signature verification model defined in the following sections.

1. CA and user certificates can contain attributes that document the conditions and assumptions under which they were created. Verifiers may simply reject all certificates and transactions that do not meet their minimum standards.

2. Attribute certificates can be signed by a user's *sponsor* to signify that their signature will be honored for official business **if** the transaction meets the requirements stated or implied by the attributes. Typically the user's sponsor will be their employer, but the model can be extended to include their bank, credit card issuer, voting bureau, video rental store, public library, or any other entity that might *accept* the user's signature. (This *authorization (sponsor) certificate* is thus the electronic equivalent of an "affidavit of legal mark" [6], as used in the context of a traditional signature stamp.)

3. Industries may develop "industry policy" statements that establish minimum requirements for signature verification. All participants would sign these multilateral agreements, to ensure that all counterparties would be bound by the encoded restrictions. Normally, authorization certificates should be required in all cases, and digital signatures would be deemed otherwise null and void in their absence. Industry-wide policies would also define relevant document types and classes.

4. There must be strict adherence to the principle that all restrictions can be enforced in an entirely automated manner (i.e., verification "on sight"), without reference to paper agreements or human interpretation. In complex and/or high-volume environments, this is an absolute requirement to give these security controls credibility in the eyes of audit and legal experts. Reference to trusted third parties should also be minimized to reduce verification latency times.

While these restrictions seem complex, they merely reflect ordinary business procedures made explicit for purposes of machine verification. Formerly, such controls were enforced inside the sponsor's computer systems *before* sending out the transaction. However, with the advent of multilateral distributed transactions, the user is normally off-line from their sponsor's system, so the verifier must enforce the sponsor's authorization model, as reflected in the attribute certificates. Once this methodology is specified, office software vendors will develop menu-driven systems to

create and manage user attributes, and the cost to user organizations will be relatively low.

6 Representing Policy Information in Certificates

It is desirable to encode information regarding to a CA's security policy into the attribute certificates of the CA and its subscribers, so that the verifier of a signature can use the information in determining whether to accept a signature as valid. In general, the CA's certificate will convey the rules a CA uses when making certification decisions, while the user's certificate will convey the information used by the CA when applying these rules.

6.1 Policy Information in CA Certificates

Attributes can indicate security policy and assurance information for a particular CA. This policy information could be inherited by subordinate CAs, allowing easy construction of security domains sharing a common policy. Policy attributes in a CA's certificate might include:

- *Liability Limitation:* The extent to which a CA is liable in the event of various problems (e.g., CA key compromise, defective binding); this might be *none*, *full liability*, or a specific monetary amount.

- *Trust Specification:* A description of which users and CAs a given CA can certify, expressed relative to the CA itself (e.g., "all subordinates"), to the DIT in general (e.g., "the subtree below Organization ABC"), and so forth.

- *Required Attributes:* A list of those attributes in the user's attribute certificates which must be verified against a transaction (and/or its context), in order for the transaction to be considered authorized. This attribute would be found in the certificate(s) of the sponsor, and allows a single authorization certificate to contain authorization attributes for use with multiple applications. Some suggested authorization attributes are defined below.

- *Allowable Name Forms:* A specification of the allowable name forms which the CA may certify. This information is held as:
  - a set of *name bindings*, which define the attributes which may be used to name entries of a given object class (i.e., the allowable RDN formats for entries of that class), and
  - a set of *structure rules*, which define which object classes may be adjacent (i.e. superior or subordinate) to each other in the DIT (i.e., the order in which object classes may be chained together to form a complete DN).

  This attribute may be used to restrict the type of entities which may sign transactions. For example, for wire transfer applications, it might be desirable to restrict signature capability to the organization itself, rather than users within the organization, since this is similar to the current mode of operation using DES MACs.

- *Cross Certificates:* It may be desirable from an efficiency point of view to allow entities (e.g., organizations) to cross-certify each other, to constrain the length of certification paths. On the other hand, it is not desirable to allow certification paths to contain arbitrary numbers of cross certificates, as it is difficult to determine the level of trust in the entity at the other end. Many certification architectures restrict certification paths to contain only one cross-certificate. To accommodate a wider range of policies, an attribute may be added to the attribute certificate associated with the cross-certificate, indicating that the cross-certifier explicitly allows the use of cross-certificates issued by the CA being cross-certified.

6.2 Policy Information in User Certificates

The following attributes may be present in an entity's attribute certificate. They represent the information verified by the CA when creating the certificate.

- *Binding Information:* The criteria used to bind the public key to the identity of the entity being certified. This includes:
  - the *method of delivery* (presented in person (or not), presented by authorized agent, by mail, etc.),
  - the *method of identification* (reasonable commercial practices, verified by trusted third party, dual control, fingerprint check, full background investigation, etc.),
  - the *identification documents* presented to the CA, and
  - the subject's *entity type* (individual, corporation, etc.).

- *Trusted Third Parties:* The names of any trusted third parties or agents involved in the binding process.

- *Roles:* It may be useful, for authorization purposes, to indicate which roles (both internal and external to the organization) a user may exercise. This is in contrast to a role certificate, which would be issued to the role and contain the names of all occupants.

- *Relative Identity:* A CA may wish to certify only a portion of the DN of an individual. In particular, the CA might disclaim liability for correctness of an individual's personal name, since their signature is binding on their organizational sponsor in any event. Consider the name:

  C=US; O=Bankers Trust; OU=Technology; { CN=Frank Sudia; T=AVP }

  The CA might certify only the validity of the organization, organizational unit, and title portions of the individual's distinguished name, which are easy to verify, while the personal name would only be "reasonably believed accurate."

  In view of the relative ease of obtaining false identity papers, this avoids the need for

prohibitively expensive background investigations. Such an identification can be relied on in an ordinary commercial setting, but not in a proceeding concerning a will or inheritance, for example.

- *Absolute Identity:* We define *relative identity* as the user's identity "relative" to their organizational sponsor. Put another way, we certify all elements of the user's "business card identity," except their personal name. As a special case, some CAs might undertake to certify the *absolute identity* of selected users, say the children of wealthy clients, diplomats, or national security operatives, almost certainly bolstered with biometric techniques. This would be rare, and is presented here only for completeness, to round out the "relative identity" concept.

7 Representing Authorization Information in Certificates

Additional attributes can convey restrictions which control the conditions under which a signature is valid. Without such restrictions, the risk of forgery is considered excessive, since an electronic signature can be affixed to literally anything, by anyone possessing the user's smart card and PIN. In the electronic environment, the normal context controls of document creation and physical delivery are weak or nonexistent.

Also, even authentic users are hardly trustworthy to undertake free-form offline commitments, so organizations will welcome the capability to positively restrict the scope of express signature authorization.

In addition to standard X.500 attributes such as a user's title, such authorization attributes might include:

- *Transaction Limits:* Controls on the value of transactions (or other documents) which an entity may initiate. The user (or card) would be restricted to originate transactions up to a certain monetary limit, or between two boundaries.

- *Cosignature Requirements:* Additional signatures which are required for a given signature to be considered valid. Quorum and weighting mechanisms can be used to construct fairly elaborate checks and balances which explicitly govern the level of trust in each user [3]. The order of required signatures may also be specified. (Note that cosigners are specified via the digests of their public keys, to reduce the propagation of personal name information.)

  The use of cosignatures allows an organization to effectively define checks and balances, and to explicitly specify the level of trust in a user. It also greatly reduces the risks from inadvertent compromise of a private key. It allows distribution of the authorization function over multiple locations and hardware platforms, with the resultant minimization of risk from access control failures on one of those platforms.

Group and role certificates may be used in conjunction with a cosignature mechanism to simplify the construction of cosignature requirements. For example, a transaction might require the signatures of three occupants of the *purchasing agent* role.

- *Document Types:* The user can be restricted to signing only such things as ordinary correspondence, purchase orders or other EDI transaction types, business contracts, specified financial instruments, etc., as defined by industry-wide policies. It will be desirable, for efficiency, to exclude large classes of transactions and documents.

- *Authorized Signatories:* An organization can indicate that only specific individuals can "sign for" the organization, similar to a standard "corporate resolution" to this effect. This might complement the document-type concept, as an additional control on signing of "corporate" document-types.

- *Geographical and Temporal Controls:* locations and time periods from which transactions are considered valid. Use of a local trusted "timestamp notary," is assumed. Such a notary would append a trusted timestamp to the originator's signature on a document, and sign the result.
  - Time-of-day restrictions would normally coincide with the work-week of the user's locale.
  - Location information would be associated with the notary, so as to restrict access to a specific network segment, typically the user's assigned work area. The "granularity" of location controls would depend on the network architecture.

- *Age of Signature:* The document is not valid unless the signature is verified within some specified time period. For high-value transactions this period would be quite short, while for more normal transactions (especially those sent via store-and-forward systems such as X.400), a longer interval would be appropriate. The time of verification would be provided using a receipt signed by a trusted timestamp service, containing, at a minimum, the recipient's name and the signature from the original transaction.

- *Preapproved Counterparties:* Restrict an entity to dealing with some small set of partners, a common requirement in dial-up home banking systems. Another way of stating this is that "free-form transfers" are forbidden. Sponsors know that in case of an error, they stand a much better chance of getting their money back from a large, solvent, creditworthy party (such as Merrill Lynch), and a much worse chance with a small, unknown, unauthorized one (such as the user's criminal confederate). Separate certificates should be issued for each counterparty, to prevent a competitor from obtaining the user's customer list (other than himself) in a single certificate.

- *Delegation Controls:* limits on the maximum authorizations a CA can specify when issuing an attribute certificate.

- *Confirm-To Requirement:* The signature is not valid unless the verifier sends a copy of the verified transaction to a third party (typically the user's sponsor or work supervisor) at a specified mail or network address, and (a) receives an accept or reject message, or (b) a specified time elapses. This is similar to a cosignature, but occurs *after* rather than before the transaction is sent. Such after-the-fact confirmation could be preferable in lower risk situations, where few transactions will be rejected and obtaining the cosignature in advance may be unduly burdensome.

A *filter* allows construction of restrictions involving multiple attributes. It is an expression involving the attributes described above. The attribute assertions are linked with the usual Boolean connectives: *and*, *or*, and *not*. For example, the sponsor might restrict a user to submitting transaction with a *type* equal to "purchase order", *and* a *value* less than $100,000. Assertions may involve a single attribute value (equality, less than, greater than, etc.), multiple values of an attribute (subset, superset, etc.), or the presence or absence of an attribute in the document.

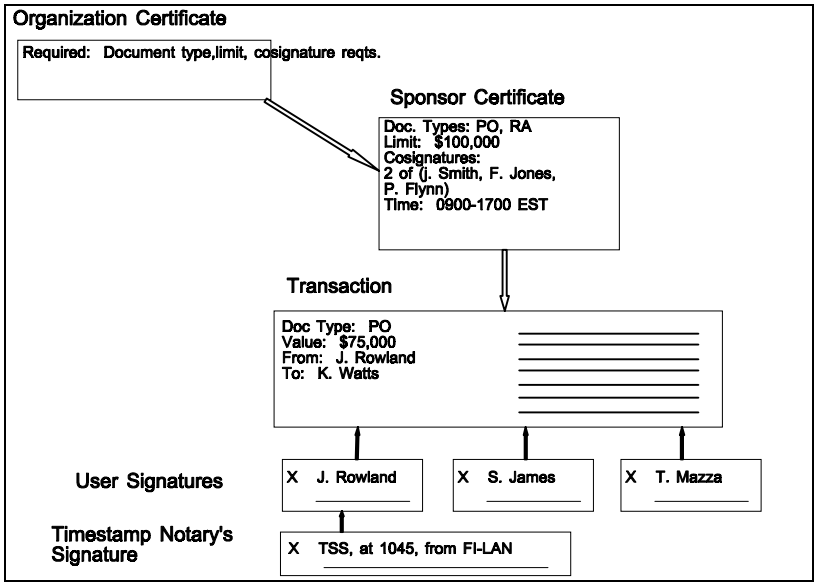

Figure 6. Authorization Model

The use of authorization attributes allows a digital signature to provide authorization as well as authentication. In such a scenario, authorization certificates (anchored by the organization's certificate) would be interpreted as *authorizing* "on sight" the transaction to which they are applied, assuming all specified restrictions are met [3].

A set of basic policies must be defined for use throughout the financial services industry (and other industries) to provide a well-defined, predictable level of service for the verification process. These policies would be agreed to on a multilateral basis by every participating firm, and could stipulate that certain of the restrictions and authorizations discussed in this section would always be deemed to be in effect, unless expressly provided otherwise.

One of the more important elements of these industry agreements would be the definition and coding of document types. This must be done on a per-industry basis, since the rules will obviously be much different for customs inspectors, aircraft inspectors, auditors, tax officials, etc. The EDI

standards promulgated by ANSI X12 may provide guidance for this effort. For the financial services industry, development of these agreements would be a logical next step after completion of the X9.30 and X9.31 standards.

8 Third Party Interactions

This section presents some useful additional features beyond those which can be provided using attribute certificates. These involve interaction between a signer and third parties of various types.

8.1 Electronic Notarization

As discussed in the previous section, there will be a need to cosign documents using a third party which is trusted to provide an accurate timestamp and/or location information. Assuming the originator provides this information in an accurate fashion leaves the door open to fraud based on, for example, pre- or post-dating of documents. An electronic "notary" would be trusted (by virtue of its CA's policies) to provide this information correctly. We can expand on the multiple signature capabilities already assumed, to provide a framework for this service.

For notarization, timestamps and location information are included as signature attributes. when countersigning the originator's signature. Individual signature structures may be detached and stored or conveyed separately from the document, if desired.

For very high-risk applications it may also be desirable to require multiple signatures on each certificate by one or more CAs, with the signatures being performed in independent cryptographic facilities (with different private keys) [2].

Various levels of service can be defined for electronic notaries, based on the level of data verification performed prior to signing (ranging from mere existence of the document, in which case notarization may be completely automatic, to human verification of document content), as well as data retention and audit capabilities.

8.2 Delegation Certificates

Any situation where one user obtains another user's smart card and PIN exposes the card to possible misuse. Yet users are highly tempted to give their cards to secretaries or co-workers, for example, when they go on vacation.

The system must therefore facilitate the issuance of "power of attorney" certificates, which allow a delegate to associate the signature of their smart card with the authority of the delegating user. Such a certificate would include, at a minimum, the name of the delegator, identification of the delegate's public key certificate, and a (short) validity period. It would be signed by the delegator. Another possibility would be for the delegate to create a new key pair for this purpose, with the new public key included in the power of attorney certificate. This would eliminate any potential

confusion between use of the delegate's private key on behalf of the delegator and on behalf of himself.

The problem of handing over smart cards can be greatly reduced by providing a workable alternative that preserves the principle of individual accountability. Wide implementation of this feature will make it practical to forbid card loans, a highly desirable goal.

The use of delegation certificates discussed above implies that the user is acting as a CA. In some cases, particularly those in which the transaction crosses organizational boundaries, there may be concern that the level of controls and auditing available with the individual user's cryptographic device (e.g., a smart card) is not sufficient. In such cases, delegation certificates could be issued by a CA, as normal authorization certificates, upon request of the delegator. (This also allows the delegation certificates to be revoked using the standard CRL mechanism.) Users' certificates might then indicate a list of possible delegates, and the delegation certificate itself would contain an attribute naming the delegator.

A user may indicate s/he is signing for another user by including a *signing-for* signature attribute, the name of the user being signed for. There must be a valid delegation certificate authorizing the signer to act for the user being signed for.

Delegation is also useful in connection with a cryptographic module in a user's personal computer. Hashing and signing a document should be a unitary operation, to prevent substitution of a false hash via software hacking. But the typical smart card lacks the computing power to hash a very long document. One solution is to let the smart card delegate this function to the cryptographic module, using a very short-lived delegation certificate (valid for a few minutes). This certificate is signed by the user's smart card, and indicates the user of the smart card has allowed the delegation [7].

8.3 Restrictions on Certificate Distribution

Users and organizations must be able to restrict the distribution of all types of certificates, because:

- They often contain confidential business information which is being shared with the verifier only for the limited purpose of signature verification;
- Users' basic privacy rights may be violated if their public keys and network addresses are published. For example, they may be flooded with unsolicited business proposals;
- There may be a general policy against giving out user IDs and public keys, as these may be used as starting points for various types of security attacks.

This functionality may be implemented in the user's attribute certificate. If the attribute is TRUE, the user/issuer is granting permission to use the certificate (and associated public key certificate)

only for signature verification; distribution or further publication is prohibited.

Other ways to specify this restriction might be to place the attribute in the organization's certificate, to publish the restriction as part of the industry-specific policy, or (in a true X.500 implementation) to use the X.500 access control list mechanism to restrict casual access to the certificate attribute.

There is some legal basis for this under copyright law, if the certificate is declared as an unpublished work with a license granted only to the named verifier. A firmer legal basis is needed.

9 Smart Card Requirements

This section describes some additional requirements on smart cards used with commercial digital signature systems.

9.1 Key Confinement and Self-Certification

The user's private key must never be allowed to leave the smart card. This is the only way we be assured that theft of the key cannot be accomplished through purely electronic means, without leaving any evidence. This is the principle of private key confinement, which is vital to the concept of non-repudiation, and hence has become an "iron law" of public key architecture.

Thus, when providing a public key to be certified, the card must attest that it is tamperproof and possesses a key confining design. Proof is provided via a certificate stating that the card is from a specific manufacturer/product line. This would be generated by the card, using a key pair reserved for this purpose. The public key of the pair must then be certified by the manufacturer. One likely approach would be to generate this "confinement" key pair during fabrication, so the corresponding certificate on the public key could be included on the card. The confinement key certifies the properties of the card using a CA associated with the manufacturer, and the card generates key pairs used to certify the user of the card, using any appropriate desired CA.

When submitting a newly generated public key for certification, the confinement private key would be used to, perhaps, countersign the certificate request data, which is signed by the newly generated private key.

9.2 Non-Decryption

Should the government require all decryption keys to be escrowed, the card should certify that it is incapable of decryption. This "signature-only" certificate can be implemented through the same mechanisms described in the previous paragraph, thus allowing the signature key to remain exempt from escrow requirements. This is vital to prevent its disclosure through possible mishandling during the escrow process, as it is doubtful whether an escrowed key retains any value for non-repudiation services.

9.3 Non-Trivial PINs

Smart cards are normally protected against unauthorized use by personal identification numbers (PINs) which are the equivalent of passwords. The PINs are changeable by the user, and must be a specified length, but typically nothing prevents the user from setting the PIN to something trivial, e.g. all 1's, or 121212. Smart card vendors should be requested to implement PIN change routines that insure non-trivial PINs, without repeating digits or obvious patterns. Making the PIN relatively long (at least 6 digits) and non-trivial reduces the chance that the card can be operated by someone finding (or stealing) it.

Cards will be required to take "evasive action", e.g., shutting down for a while, or even erasing private keys, if too many bad PINs are entered.

Support for a 6-digit PIN requirement can be found in ANSI X9.26 [4], which sets forth the "one-in-a-million" standard. A login mechanism may be considered secure if, among other things, an attacker has no more than a one-in-a-million chance of guessing the correct password, and the system takes evasive action to prevent repeated guessing.

9.4 Biometrics

Extensive work is being done in the areas of voiceprint and fingerprint identification, as a supplement to PINs. We expect to see commercial solutions within the next 12-18 months.

While the rates of false positives and negatives still must be reduced, the main problem lies in securing the biometric input device and its data channel, so they are immune to capture and replay of the biometric data. This is not a problem if the biometric device is embedded in a concrete wall, for example in an ATM or door access system. It remains a serious problem in typical commercial office settings however. Eventually, the card, the PC, and biometric input device will each contain (or be) tamperproof cryptographic modules, which can certify themselves and establish secure channels with each other.

9.5 Audit Trails in User Tokens

Smart cards should maintain an internal log of recent actions, containing at a minimum, a timestamp, transaction amount, type code, and signature. This information can be compressed into 40 or so bytes, so a 400 record circular log would consume around 16K bytes. This log would be dumped only on receipt of a signed request from the card issuer, received over a secure channel.

This control mechanism will deter forgery, reduce the damage that can be caused by a forger, and allow unauthorized or questioned transactions to be investigated more quickly and easily. Since most or all transactions occur off-line from the issuer, the card is the best witness of its own actions.

References

[1] CCITT, "X.509: The Directory: Authentication Framework," April, 1993.

[2] ANSI X9F1, "X9.30 Part 3: Certificate Management for DSA," July, 1993 (draft).

[3] Fischer, A., "Electronic Document Authorization," Proceedings of the 13th National Computer Security Conference, 1990.

[4] ANSI, "X9.26: Financial Institution Sign-On Authentication for Wholesale Financial Transactions", 1990.

[5] Bulletin boards for lawyers. (ABA EDI Workgroup paper).

[6] "Affidavit" terminology is derived from: Robert Jueneman, "Limiting the Liability of CAs and Individuals Regarding the Use of Digital Signatures," presented to the ABA Section of Science and Technology Certification Authority Work Group, July 2, 1993.

[7] This idea is expressed in Digital Equipment's DSSA system. See, for example: Gasser, M., A. Goldstein, C. Kaufman and B. Lampson, "The Digital Distributed System Security Architecture," Proceedings of the 12th National Computer Security Conference, 1989, or: Gasser, M. and E. McDermott, "An Architecture for Practical Delegation in a Distributed System," Proceedings of the 1990 IEEE Symposium on Security and Privacy.

[8] RSA Data Security, Inc., "PKCS #7: Cryptographic Message Syntax," 1993.

[9] ANSI, "X12.58 version 2: EDI Security Structures," July, 1993.