Spring 2-2011

Interoperable Credentials Management for Wholesale Banking

Glenn Benson
JPMorgan Chase & Co

Shiu-Kai Chin
Syracuse University, skchin@syr.edu

Sean Croston
JPMorgan Chase & Co

Karthick Jayaraman
Syracuse University

Susan Older
Syracuse University, sbolder@syr.edu

Interoperable Credentials Management for Wholesale Banking

Glenn Benson[‡], Shiu-Kai Chin[†], Sean B Croston[‡], Karthick Jayaraman[†], and Susan Older[†]

[‡]JPMorgan Chase & Co
glenn.benson@jpmchase.com
sean.b.croston@jpmchase.com

[†]Syracuse University
skchin@syr.edu
kjayaram@syr.edu
sbolder@syr.edu

Abstract

A gap exists between wholesale-banking business practices and security best practices: wholesale banks operate within the boundaries of contract law, while security best practices often relies upon a benevolent trusted party outside the scope of straightforward contracts. While some business domains may be able to bridge this gap, the ultra-high-value transactions used in business-to-business banking substantially increase the size of the gap. The gap becomes most apparent when regarded from the perspective of interoperability. If a single user applies the same credential to sign high-value transactions at multiple banks, then the trusted-party model becomes overly cumbersome and conflicts with an acceptable concept of liability.

This paper outlines the business complexities of wholesale banking and proposes a solution called Partner Key Management (PKM). PKM technology manages the credentials required to authenticate users and sign transactions. This paper presents PKM technology by describing an interoperable protocol, requisite data structures, and an interoperable XML definition. The paper uses formal methods to demonstrate a security equivalence between revocation options within PKM against the security offered by the traditional Public Key Infrastructure (PKI), a technology that features the benevolent trusted party.

1 Introduction

Wholesale banking is a high-volume banking service offered to corporations, institutions, and governments. Wholesale banking resembles retail online banking, but it provides more complex services tailored for the corporate needs. Typically, customers use their services on a daily basis, execute a relatively large number of transactions, and often release transactions of high dollar amounts. Cash managers are the employees within the corporate customers' treasury departments who are authorized to represent their companies' daily financial management. The cash managers handle their companies' accounts receivables and payables by monitoring the incoming transactions and releasing transactions from their own bank accounts.

A large international corporation may have as many as ten or more banks distributed throughout the world. Although the need to hold bank accounts at multiple banks may be apparent from a financial perspective, the cash manager currently faces relatively poor security ergonomics. Each bank typically offers its own credentials for authentication and digital signatures. A single user may have a few USB token-based certificates, some one-time password tokens such as those offered by SecurID[1] or Vasco[2], multiple passwords, and possibly some proprietary credentials. Because wholesale banking usually does not offer credential interoperability, the cash managers have no choice but to manage a desk drawer full of authentication and signature credentials.

The obvious solution is to ask wholesale banks to cooperate upon a common standard that supports credential interoperability. A credential is interoperable among a set of partners, if and only if all the partners have the same interpretation of information furnished in the credential. The current industry best-practice technology is the Public Key Infrastructure (PKI) [1], but PKI is an ill-fit for interoperable wholesale banking. One of the fundamental drawbacks of PKI is its focus on a benevolent trusted party (i.e., a certificate authority): the concept of a benevolent trusted party poorly reflects traditional contract law. Furthermore, PKI has been designed to also support identity interoperability, which requires that all partners view the same attributes and share the same process for accepting and processing an identity. As a result, PKI forces a centralized governance model, which does not address business needs of wholesale banking.

In general, PKI cannot easily address the four most important high-level business requirements for interoperable wholesale banking:

1. *Build a comprehensive liability framework that governs fraudulent or mistaken transactions.*

   An environment that permits transactions of hundreds of millions of dollars[3] leaves little room for liability ambiguity in the business model. No bank wishes to be liable for transactions conducted at another bank, and no party will agree to unlimited liability for another party's operations. Furthermore, it is extremely unlikely that a certificate authority would have the resources to indemnify the banks, if the certificate authority were found to be in error.

2. *Delegate credential administration to the corporate customer.*

   At present, banks are often forced to develop additional security techniques as a direct consequence of using PKIs. For example, both South Korea and Brazil certify interoperable PKI providers and mandate that wholesale banks subscribe to the nationally certified PKIs [14]. The Brazilian PKI exhibits inherent shortcomings. For example, if a user loses his own certificate, then he can contact the central infrastructure to request certificate revocation. However, the Brazilian PKI providers do not enable an employer to revoke an employee's certificate with cause (e.g., if the employee has a gambling problem). As a result, banks must build their own additional controls to

[1] `http://www.rsa.com/node.aspx?id=1156`

[2] `http://www.vasco.com`

[3] In 2009, for example, the wholesale-banking division of J.P. Morgan averaged more than US$58 Million per second in value-bearing transactions and as much as US$5.1 Trillion of money movement in a single day.

cover for limitation of the PKI.

A better solution is a system that maintains a list of authorized administrators who may speak for a corporation with regards to managing individual users' certificates.

3. *Create an agile marketplace that encourages global competition.*

   The global banking environment should not require each bank to simultaneously manage multiple connections, periodically test with multiple infrastructures, maintain complex redundant disaster recovery environments, and otherwise suffer the burdens when interoperable infrastructures cannot themselves mutually interoperate. The interoperable solution may require a technological standardization process, but the standards should not extend into business practices or liability models.

4. *Implement a scalable solution that operates in an international banking environment.*

   The solution should reflect existing contract law and generally accepted business practices; it should not require an industry to re-engineer its business processes to meet the expectations of security technology. Furthermore, the solution should permit adequate autonomy to banks and their customers.

   Unfortunately, adopting PKI often requires the standardization of business practices and national regulations. PKI's governance model extends into a revocation model, timestamping standards, and a liability model. Consequently, universal adoption of a common PKI or bridged PKIs [15] may potentially require international treaties at the level of national governments and expensive remediation of individual banks' business practices. The task of modifying and standardizing business practices and national regulations is extraordinarily difficult and time-consuming. If the industry needs to wait for difficult concepts such as a globally regulated liability model or a globally-bridged PKI across all major nations, then we will not see interoperable credentials for a very long time.

A final example of the ill-fit between PKI and the wholesale-banking industry appears in the use of identities. PKI supports identity interoperability (a non-need of the banking industry) by establishing a distinguished namespace. Conforming to this distinguished namespace imposes additional overhead upon the wholesale-banking industry without addressing any actual business need: banking already has its own business-oriented concept of bank-identification codes (e.g., bank routing numbers), and unique bank numbers. Although each bank needs to identify its users, the banking industry does not require inter-banking identity interoperability, i.e., no wholesale bank is required to observe the same identity attributes as some other wholesale bank.

In this paper, we propose Partner Key Management (PKM) as a solution that supports the wholesale-banking business requirements without imposing technological requirements that extend beyond business needs. Specifically, PKM provides the following in direct response to the wholesale-banking business requirements:

1. A means of communication whereby a corporation and a bank securely agree upon the credentials used to prove the identity of each user. The protocol does not transfer liability among multiple banks.

2. A practical method by which a corporation may appoint multiple employees to act as the corporate representative governing its users' credentials.

3. A flexible credential-revocation model that reflects contract law as opposed to benevolent trusted third parties

4. An ability for corporations and banks to bilaterally agree upon revocation, timestamp, and liability models without mandating complex intercorporate, interbank, or international business agreements.

J.P. Morgan operates PKM for its host-host wholesale banking solution offered to its customers. Subsequent to J.P. Morgan's implementation, SWIFT [4] deployed PKM in its product called the SWIFT 3SKey[5].

The remainder of this paper is organized as follows. In Section 2, we introduce the core concepts underlying PKM: the credential-registration process, the credential-validation protocols, and the Partner Key Policy Statement (PKPS) that supports bilateral agreements between corporations and banks. In Section 3, we describe in more detail the XML structures and the protocols that support these core PKM concepts. In Section 4, we use a formal logic to demonstrate an equivalence and means of comparison between the security afforded by a PKI and PKM. We give an overview of related technologies in Section 5 and conclude in Section 6.

2 Partner Key Management Concepts

In this section, we introduce the core concepts that underlie Partner Key Management (PKM). We defer to Section 3 a description of how these core concepts can be implemented and used in practice.

2.1 Credential Registration

PKM includes a three-step credential-management process, which is illustrated in Figure 1. In Step 1, the user (e.g., the corporate cash manager) obtains a credential, from either the corporation itself or a third-party credential provider. In Step 2, the user contacts one of its banks with a request to register the credential (i.e., to establish an association between the userid and the registered credential). In Step 3, the bank determines whether or not to register the submitted credential. The basis for this determination depends upon the individual bank's policy: typically, a security administrator at the corporation will contact the bank for verification that the credential in fact should be registered.

A corporation may permit its employees to register credentials in all of the banks with which the corporation conducts business. If an employee chooses to abuse his or her privilege by registering a credential at an unapproved bank, then the corporate security administrator should not approve the unauthorized registration in Step 3. The multi-bank registration process realizes the goal of credential interoperability, because the user may employ

[4] SWIFT is a financial services leader with more than 9,000 banking organizations, securities institutions and corporate customers in 209 countries

[5] http://www.swift.com/products/3skey. SWIFT and 3SKey are trademarks of the S.W.I.F.T. SCRL.

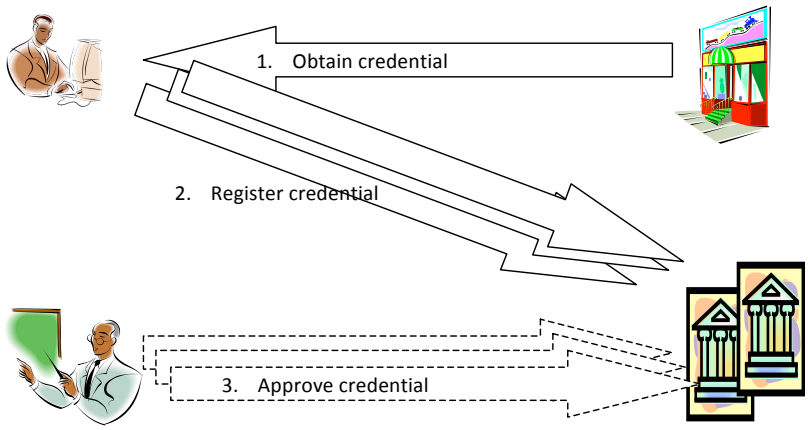

Figure 1: Credential registration in PKM

.

the same credential with multiple institutions. However, the protocol does not provide the unneeded identity interoperability, because no requirement exists forcing all the banks to recognize a single userid or impose a universally recognized distinguished name.

PKM itself does not require any of the three steps to be secured in any particular way. Rather, each bank has the freedom to impose its own security requirements and protocol without restriction of an interbank standard or governance model. Therefore, no financial institution suffers due to another financial institution's respective security shortcomings, because each financial institution's registration process has no dependency upon any other financial institution. Among the vehicles that a financial institution may potentially employ to secure the PKM steps are a manual bootstrap procedure and signed approvals:

- *Manual bootstrap:* The user sends its credential to the bank using a trusted physical courier. The user sends additional wet-ink signed documentation authorizing the credential. The financial institution and user may also employ other steps to fully authenticate the transaction. For example, they may use phone calls or fax transmissions with the user or other administrators.

- *Signed approvals:* At the conclusion of manual bootstrap, multiple administrators at a corporation may possess registered and authorized credentials. These administrators may use those credentials to contact the bank securely in Step 3. In other words, the administrators may use previously registered credentials to approve other users' new credentials.

Furthermore, PKM imposes few constraints upon the type of credential. Among other possibilities, the following credentials are all supported:

- *Certificates issued by a certificate authority:* A user may obtain a certificate from a recognized certificate authority (CA). PKM does not impose any limitations upon this process. However, a financial institution and corporation may optionally choose to impose their own restrictions.

- *Self-signed certificates:* A user may register a self-signed certificate with the bank. A self-signed certificate is a degenerative case of Figure 1, because the user operates as its own credential distributor. However, to ensure security, the corporation or the bank may wish to ensure that the administrator in Step 3 is different from the user. As long as the bank trusts the approval mechanism, a PKM-registered self-signed certificate provides as much security as a CA-issued certificate.

- *One-time passwords:* A one-time password normally has a unique serial number known by both the client and the server. Both parties use this serial number to uniquely identify the credential. The user registers the serial number in Step 2 along with his or her userid. A one-time password can be the authentication component of a digital signature; see [4] for details.

This paper focuses on credentials that participate in digital-signature processes in wholesale banking. However, PKM may also register credentials that do not contribute to digital signatures or that are used outside the financial services sector, such as a restriction of the IP address from which a user may connect or a registration of a SIM card for a mobile phone. At the conclusion of the credential registration process, the credential *speaks for* the user. That is, when the bank receives a transaction signed by the credential, the bank understands that the user authorizes the transaction's execution. PKM also assumes an analogous un-registration process. An authorized representative of the corporation may instruct the bank to stop accepting a previously registered credential.

PKM directly addresses the liability and credential-administration business requirements. Because no bank relies upon another bank or other entity's registration process, the inter-bank liability issue evaporates. As for credential administration, the inherent limitations that affect a benevolent trusted party do not apply to banks. The banks have the freedom and autonomy to implement any authorization process that they choose, and each bank may require as many or as few Step 3 authorizing parties as it wants. Since each bank needs to closely interact with its own customers anyway, in order to manage user privileges, the additional administration burden imposed by PKM may be minimal.

2.2 Partner Key Policy Statements

Banks participating in the PKM model publish one or more XML [5] documents called the Partner Key Policy Statement (PKPS), which comply to the WS-Policy [18] XML schema. A PKPS defines how a corporation and a bank agree to work together, as governed by their mutually agreed security operating rules. The corporation and the bank have the freedom to impose almost any conditions to which they mutually agree, provided that the conditions do not require unsupportable programming logic.

Figure 2 contains a flowchart detailing the steps for validating a PKPS. A user's transaction request comprises a signed transaction document and a signed PKPS. The bank compares the incoming PKPS against its PKPS repository to find a match. If the bank does not find a matching PKPS, then it rejects the transaction and returns an error to the user. Through an offline process, the bank and the user must correct their misunderstanding before the user can submit any more signed transactions. If the bank finds a matching PKPS, then the bank validates the signatures on the PKPS to verify that the authorized signatories signed the PKPS. If signature validation succeeds, then the bank processes the transaction

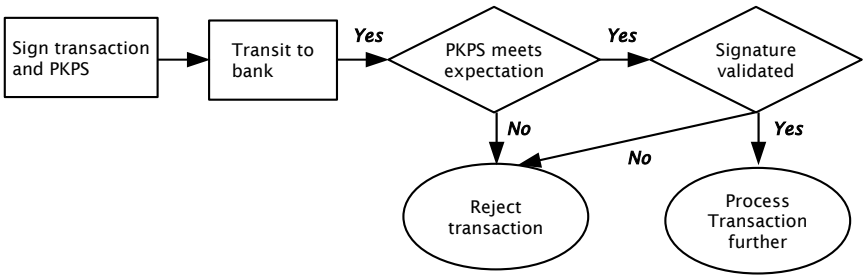

Figure 2: Signature processing in PKM

further including transaction signature validation. In effect, a bank considers a PKPS validated if and only if the bank finds a matching PKPS and the PKPS is signed by authorized signatories.

A given PKPS specifies a collection of policies to which the user and the banks must agree. The PKPS may include any of the following specific policies:

1. *Credential Media:* The definition of the credential media may mandate a particular FIPS-140-2 [12] level of protection.

2. *Credential Provider:* This item contains the list of credential providers to which the corporation and the bank mutually subscribe. Example providers are third party trusted providers, self-signed certificates, or the corporation's or the bank's own provider.

3. *Revocation:* The revocation definition describes the type of permissible credential revocation mechanism, such as a certificate revocation list (CRL) or an online certificate status protocol (OCSP) [17]. The revocation definition also describes the party responsible for enforcing credential revocation and any specific usage practice. Subsection 2.3 presents details.

4. *Timestamp:* The timestamp definition defines timestamp rules and the timestamp provider, if any. The timestamp definition may specify either a real-time threshold value (i.e., a limit on how long past the timestamp a signature can be validated) or a real-time constraint.

5. *Signature Policy:* The PKPS can specify the number of signatures required for a specific type of transaction, as well as the roles of signatories. For example, a signature policy may require both an individual signature and a corporate "system" signature to be present.

6. *Credential Technology:* The credential-technology section specifies the standards and agreements that must be used, such as X.509 certificates [10] or PGP certificates [6]. PKPS additionally opens the possibility of technology advancements by allowing banks and customers to agree to use technologies that have not yet been submitted for global standardization. For example, the J.P. Morgan wholesale bank customers use the Portable Security Transaction Protocol (PSTP) to sign their transactions [4].

Figure 3: Islands of interoperability

Figure 3 illustrates possible scenarios where one or more entities recognize a single PKPS. It also shows a natural progression that industries may take in terms of supporting credential interoperability. In Scenario (a), a single bank defines its own, unique PKPS. The bank informs its customers that it rejects all incoming signatures that contain either no PKPS or a PKPS that differs from its expectation. The bank's customers benefit from a limited form of credential interoperability: they can use their credential at each bank that handles PKM, even if different banks do not recognize the same PKPS. It is possible that customers may demand further interoperability of any or all of the items covered by a PKPS. In reaction to market pressure, a group of banks may decide to band together, harmonize their differences, and agree to recognize a common PKPS, as in Scenario (b) of Figure 3. We call this group of banks an "island of interoperability," because interoperable governance exists only within the island. Scenario (c) reflects a larger, nationwide island of interoperability, in which national governments (such as Korea and Brazil) mandate a credential governance model across all wholesale banks serving their nation. Proceeding further, like-minded nations such as the Nordic region may band together to form a very large island, as in Scenario (d). The global interoperable governance of Scenario (e) is unlikely in the near future, but we may consider it as a distant, albeit elusive possibility. PKM allows each industry to progress toward Scenario (e) at its own rate. Market pressures—as opposed to governmental fiat—dictate the relative speed at which the banks must work toward improved interoperability.

Brazil, Korea, and the Nordic nations are all examples of nations or regions that have large-scale interoperable PKIs today. If these PKI regions were to upgrade to PKM, then they could potentially extend their reach beyond the current boundaries while addressing their own inherent deficiencies in liability handling or user administration. Alternatively, a bank that adopts PKM could work in any of these regions by accepting the region-specific certificates through the PKM process.

2.3 Example Revocation Models

One aspect of a PKPS that merits special attention is revocation. We present four example revocation models, which are illustrated in Figure 4.

Receiver validation: The receiver-validation model is typically used in a PKI. First, Alice submits a signed transaction to the bank. Upon receipt, the bank validates the certificate employed in the signature against a CRL or OCSP responder managed by the certificate provider.

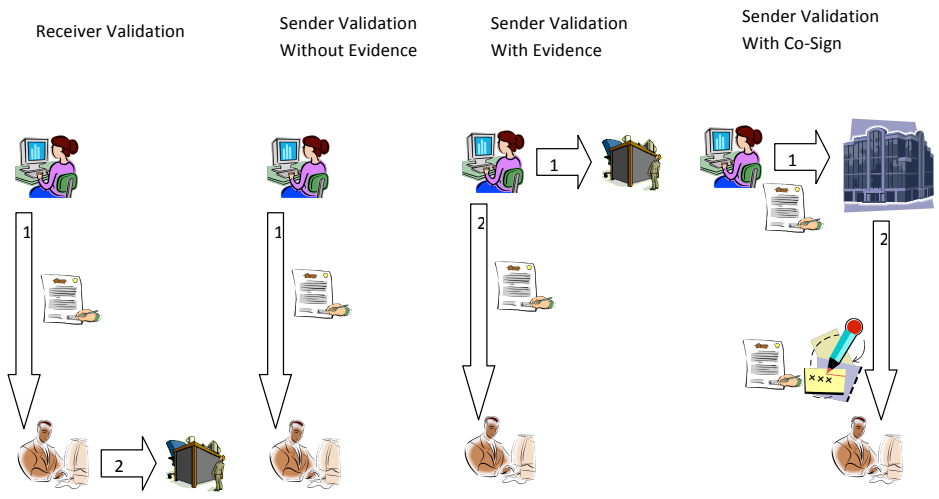

Figure 4: Validation models

Sender validation without evidence: Alice submits signed transactions to the bank, but the bank performs no revocation check other than looking to see if Alice's credential has been registered but not unregistered.

Sender validation with evidence: Alice submits her certificate to an OCSP responder, and obtains a response signed by the OCSP responder. Alice signs both the transaction and the OCSP response, which she then submits to the bank. The bank validates both Alice's signature and the OCSP responder's signature. If the bank finds no error, then the bank accepts the transaction.

Sender validation with cosign: A signer's signature must have an accompanying cosignature. Alice first signs a transaction then routes the signed transaction to a central corporate facility for a cosignature. The central corporate facility validates Alice's identity and ensures that her credentials are current and valid before executing the cosignature.

Each bank has the opportunity to allow any of the example models or to build its own revocation model. PKM permits governmental autonomy: multiple banks can all accept the same credential from Alice while simultaneously establishing their own governance rules (e.g., required PKPS structures or revocation models).

Receiver validation is the most common revocation type in the industry today, because most PKIs support it. However, receiver validation is not a good technique to address the agile-marketplace business requirement. If each bank must connect to every CA in order to validate the certificate of every signed transaction, then the availability of a bank is no better than the availability of the CAs. Certificate Revocation Lists (CRLs) may be better than OCSP because the bank can shield itself from minor network disruptions through caching. Nevertheless, if a single bank must connect to many different infrastructures, then the bank cannot provide an adequate Service Level Agreement. Customers might be unsympathetic if a bank were to blame its unexpected downtime upon a CA's lack of service. Furthermore, the cost of connecting to each infrastructure may significantly impede the global mar-

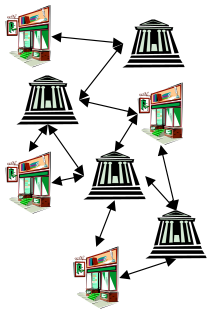

Figure 5: Bilateral banking agreements

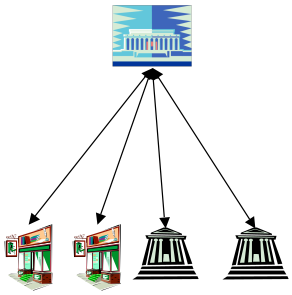

Figure 6: Hierarchical agreements

ketplace. A connection requires not only development cost, but periodic testing, disaster-recovery planning, audits, and maintenance. These costs would discourage banks from accepting customer requests to use the customer's chosen infrastructure, even if that infrastructure were well-behaved.

In contrast, all three sender-validation models can optimize agility. Each corporation needs to build an on-line connection to either one or zero infrastructures, depending upon the variant of the model. The banks do not need to connect to any infrastructures. In fact, the wholesale banking division of J.P. Morgan uses a manual sender validation method to register the correct keys from partners. In the manual method, a partner sends its correct keys and the evidence of their validity (signed with plain signatures) using courier services or fax to J.P. Morgan to register the keys. The PKM protocol provides a digital alternative for achieving the same intent. Furthermore, the security of sender validation is as good as receiver validation – see Section 4 for details.

3 Partner Key Management Technology

The wholesale-banking business operates through a network of contractual agreements between corporations, banks, and other financial institutions. Figures 5 and 6 together illustrate the difference between the network of bilateral contractual agreements that typifies wholesale banking and the hierarchical agreements offered by PKI's benevolent trusted-party model.

In this section, we describe the technologies that allow PKM to support this network of bilateral contractual agreements. We begin by introducing the signing mechanisms. We then describe how signatures are used in the four primary revocation models. Finally, we present the overall structure and components of the Partner Key Policy Statement (PKPS).

3.1 Signature Binding of PKPS

A PKPS is an XML document that has no inherent protection against unauthorized modification and has no concept of ownership. Consequently, every PKPS should reside within the context of at least one digital signature. The format of the digital signature is outside of the scope of the PKPS specification; however, because the PKPS syntax is typically XML, one would expect the most applicable signature format to be XMLDSIG [3]. Although the

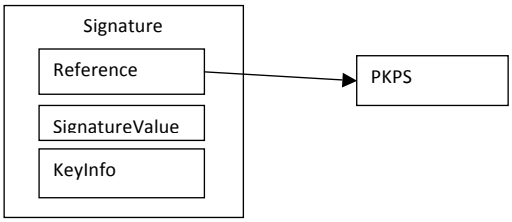

Figure 7: XMLDSIG signature binding

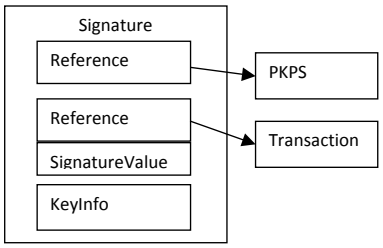

Figure 8: Single signature on a PKPS

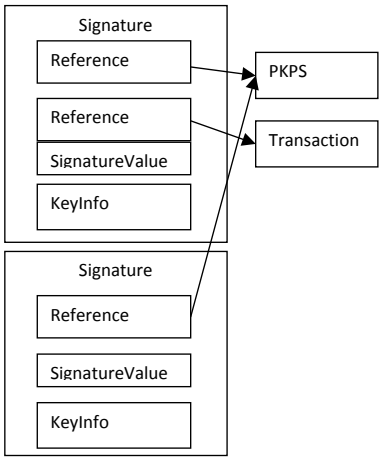

Figure 9: Multiple signatures on a PKPS

PKPS does not constrain whether its associated signatures use the XMLDSIG detached or non-detached formats, we suspect that the detached format may be best for most use cases.

Figure 7 illustrates an XMLDSIG signature that covers a PKPS. In accordance to the XMLDSIG standard, the reference contains a digest of the referenced document (which in this case is a PKPS), and the signature value covers all the references. Thus, if an adversary were to attempt to modify a PKPS, then either the reference's digest would fail to validate or the SignatureValue computed over a substituted digest would fail. The KeyInfo is an optional XMLDSIG element that provides the keys needed to validate the signature.

Multiple options exist for signatures, as illustrated in Figures 8 and 9. In Figure 8, a single signature covers both a PKPS and a transaction document, which describes the details of the transaction requested. In Figure 9, multiple signatures cover a single PKPS, and one of the signatures also covers the transaction document. In the next subsection, we look at revocation models that employ various signature strategies.

3.2 Revocation Models

We introduced four revocation models in Subsection 2.3. We now explain how signatures and signing mechanisms support these different models.

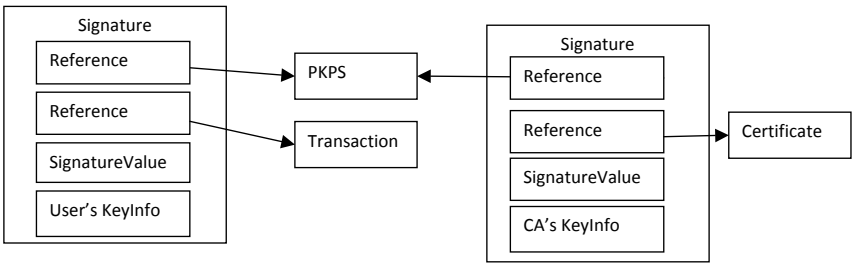

Figure 10: Signature structure for receiver validation

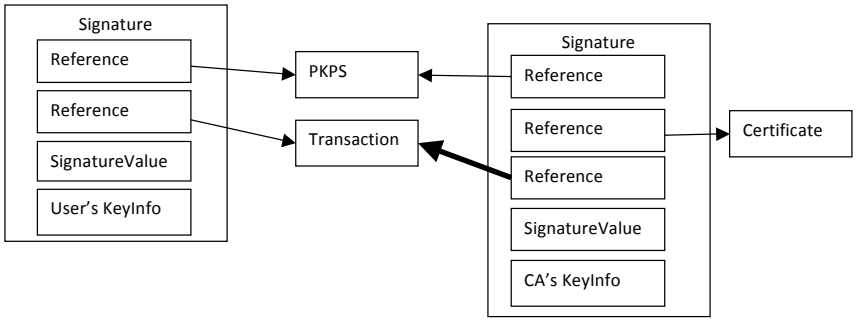

Figure 11: Signature structure for sender validation with evidence

Receiver Validation Figure 10 illustrates the data structure for the receiver-validation model. The signature on the left of the data structure is the XMLDSIG executed by the user: the user identifies the transaction document that he or she wishes to sign and then executes the signature over both the transaction document and the PKPS. The signature on the right illustrates the CA's signature: the CA signs the PKPS and the user's certificate, but not the transaction-level document. Consequently, the CA does not need to wait to see the transaction document and can pre compute its signature. In comparison, the receiver-validation data structure is similar to the data structure that one would use in a standard PKI with X.509 certificates. The advantage is that XML relieves the X.509 certificate of the burden of handling certificate extensions. Instead, the data structure may place the information that one would normally find in a certificate extension into the PKPS (perhaps leveraging an extension of the PKPS schema when necessary). The advantage is that the PKPS uses a more modern XML format, as opposed to the X.509 extension's use of the antiquated ASN.1 syntax.

Sender Validation without Evidence The sender-validation-without-evidence model requires only the left half of Figure 10: the user signs both the PKPS and the transaction document. No CA signature is required in this model.

Sender Validation with Evidence The sender-validation-with-evidence model adds additional signature coverage to the receiver-validation model. The bold arrow in Figure 11 high-

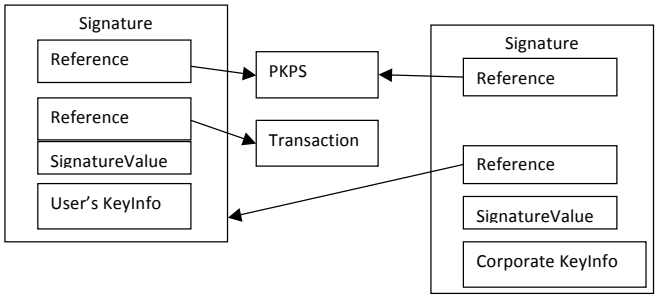

Figure 12: Signature structure for sender validation with cosign

lights the sole conceptual difference between the receiver-validation and sender-validation-with-evidence models: the transaction is signed by both the corporate user and the OCSP responder. The user first signs both the PKPS and the transaction document, as shown on the left of Figure 11. The user then sends both her certificate and a message digest of the transaction to the OCSP responder. If the OCSP responder considers the certificate to be currently valid, then the OCSP responder signs the message digest and send this signature to the user. The user may add this signature to the data structure and then submit the entire structure to the intended receiver. In order to optimize operations, the CA may elect to use different keys to sign the certificates and OCSP responses. In this case, the data structure of Figure 11 would be more complex, but it would serve the same purpose.

Sender Validation with Cosign The sender-validation-with-cosign data structure appears in Figure 12; and Figure 13 presents the process for creating a sender-validation-with-cosign signature. The three-step signature process starts with a user who signs a transaction document and then sends the signature to a centralized automated validator in the corporate data center. In the second step, the automated validator consults human-resource records or other facilities to verify the validity of the user's credential. If everything checks out, then the automated validator countersigns to indicate the current validity of the user's credential and forwards it to the bank. In effect, the corporation asserts a limited scope acknowledgment: the corporation agrees that the user signed the transaction with a valid key. However, the corporation does not necessarily view or acknowledge the transaction details. In the third step, the bank validates both signatures. The user's signature indicates an agreement to the transaction document; the automated validator's signature indicates an agreement to the current validity of the user's credential.

The purpose of this revocation model is to replace trust in a benevolent third party with contract law. If the corporation were to lie by signing a transaction inappropriately, then the corporation would be in breach of contract. Furthermore, a corporate lie would be against the corporation's best interest, because it would permit signatures using certificates that should no longer be valid.

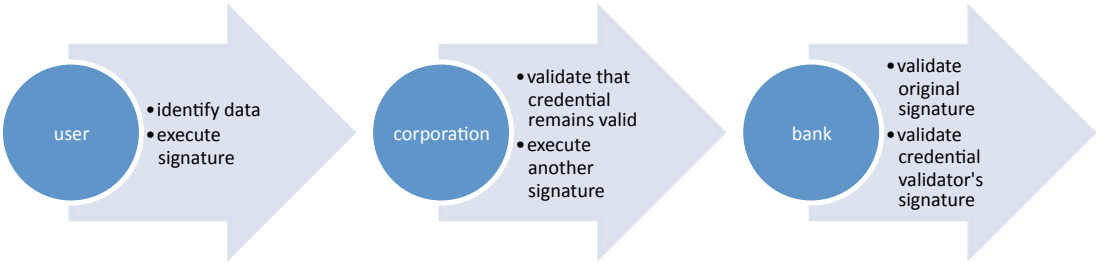

Figure 13: Three-step signature process for sender validation with cosign

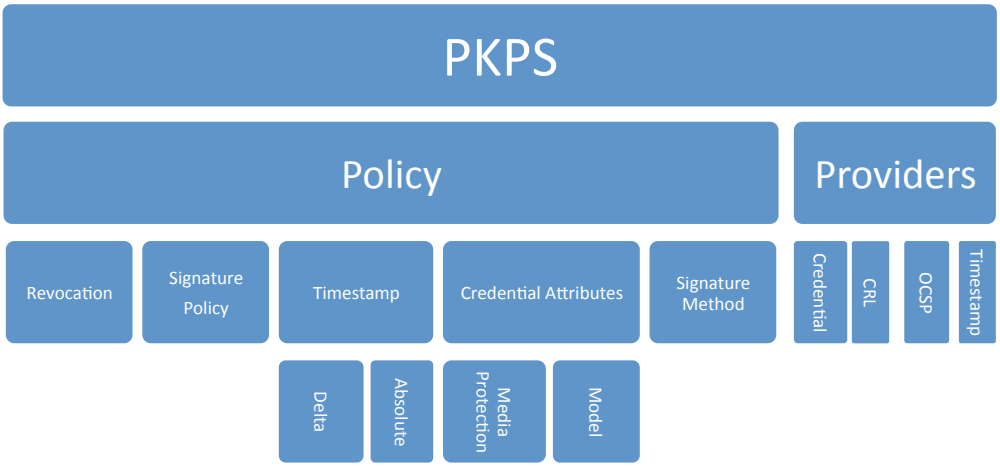

Figure 14: PKPS Structure


```
<xs:element name="Revocation">
        <xs:complexType>
                <xs:choice>
                        <xs:element ref="ns1:receiver-validation" />
                        <xs:element ref="ns1:sender-validation-without-evidence" />
                        <xs:element ref="ns1:sender-validation-with-evidence" />
                        <xs:element ref="ns1:sender-validation-with-cosign" />
                </xs:choice>
        </xs:complexType>
</xs:element>
```

Figure 15: XML schema for revocation elements

3.3 PKPS Syntax and Semantics

We now turn our attention to the structure of the PKPS, which is illustrated in Figure 14. The PKPS has two major sections: Policy and Providers. The Policy section describes specific agreements between the parties that observe the PKPS. The Providers section describes any requirements for specific third parties referenced by any of the policies. All sections of the PKPS are optional.

Revocation The Revocation element specifies the type of credential revocation model to be used. Figure 15 contains the XML schema for the Revocation element.

Signature Policy The Signature Policy element describes the roles required for a signature. For example, consider the sample signature policy given in Figure 16. This policy indicates that a transaction governed by the PKPS must meet one of two alternatives: (1) it must contain signatures by two individuals with the Cash Manager role and another individual with the Corporate role, or (2) it must contain signatures by separate individuals in the Cash Manager, Senior Manager, and Corporate roles. If a particular transaction document does not contain all of the required signatures, then the parties must consider the transaction document to be unsigned.

One of the purposes of the Signature Policy is to protect against phishing and other social engineering attacks. Suppose an authorized party were to accidentally lose his or her credential. The Signature Policy limits damage by forcing the signature to be invalid until the adversary breaks the security of the other required signatories.

Of course, the corporation and the bank each have interest in ensuring correct role assignments to individuals. However, the role assignment is outside the scope of PKM. In other words, PKM does not need to standardize on a particular role assignment protocol or role names; rather, each corporation and bank have the flexibility to agree upon their own rules and protocols bilaterally. Figure 17 contains the signature policy schema.

A degenerative signature policy is one that requires no signers whatsoever; it serves as a pure authentication event without a data signature. Suppose, for example, that the bank asks the user to sign a bank-generated random number at login time to protect against playback attacks. The user responds by concatenating the random number to a PKPS with the degenerative signature policy. Doing so protects the user against a case where an adversary tries to cheat by sending a message digest of a transaction instead of a random number.


```
<ns1:SignaturePolicy>
 <wsp:Policy>
  <wsp:ExactlyOne>
   <ns1:Roles>
     <ns1:Role>Cash Manager </ns1:Role>
     <ns1:Role>Cash Manager </ns1:Role>
     <ns1:Role>Corporate </ns1:Role>
   </ns1:Roles>
   <ns1:Roles>
    <ns1:Role>Cash Manager </ns1:Role>
    <ns1:Role>Senior Manager </ns1:Role>
    <ns1:Role>Corporate </ns1:Role>
   </ns1:Roles>
  </wsp:ExactlyOne>
 </wsp:Policy>
</ns1:SignaturePolicy>
```

Figure 16: Sample signature policy

```
<xs:element name="SignaturePolicy">
        <xs:complexType>
                <xs:choice>
                        <xs:element ref="wsp:Policy"/>
                        <xs:element ref="wsp:PolicyReference"/>
                </xs:choice>
        </xs:complexType>
</xs:element>
<xs:element name="Roles" type="ns1:RolesType"/>
<xs:complexType name="RolesType">
        <xs:sequence maxOccurs="unbounded">
                <xs:element name="Role" type="xs:string"/>
        </xs:sequence>
</xs:complexType>
```

Figure 17: XML schema for signature policies


```
<xs:element name="Timestamp" type="ns1:WSPolicy"/>
<xs:element name="Threshold" type="ns1:TimestampType" />
<xs:element name="Absolute" type="ns1:TimestampType" />
<xs:complexType name="TimestampType" >
        <xs:sequence>

                <xs:element name="TZ" type="xs:string"/>
                <xs:element name="Hours" type="xs:integer"/>
                <xs:element name="Minutes" type="xs:integer"/>
                <xs:element name="Seconds" type="xs:integer"/>
        </xs:sequence>
</xs:complexType>
```

Figure 18: XML schema for timestamps

The user would win any subsequent dispute by showing that his signature covers only a degenerative signature policy.

Timestamp Figure 18 contains the XML Schema for the timestamp. Two types of timestamp policies exist: absolute and threshold. An absolute policy specifies the cut-off time on a particular day, after which the bank refuses to accept the signed transaction. In this case, the Hours, Minutes, and Seconds in the XML represent an absolute time deadline on the day it is received.

A threshold policy specifies an upper bound for the difference between the claimed signing time and the time of receipt. In this case, the Hours, Minutes, and Seconds in the XML provide the upper bound for this difference. The means of validating the claimed signing time is outside PKM's scope, but either of the following examples might be dictated by the specific local environment:

- In online banking, a user points a browser at a web page. The bank knows that the user must have executed the signature sometime after receiving the web page and before the bank received the signature. Thus, the claimed signing time must be between these points in time.
- The user may append a timestamp produced by a third-party trusted timestamp provider. If the bank recognizes this same timestamp provider, then the bank can validate the timestamp.

Credential Attributes The credential-attributes section specifies two constraints: FIPS level and model. The purpose of this section is to characterize the level of trust of the credential media used to carry the credentials. The FIPS level is the level of assurance that the private cryptographic key does not leak off it media; the FIPS-140-2 [12] is an NIST[6] standard that provides four increasingly stringent certification levels for cryptographic modules. A bank that wants its customers to employ the highest level of security should mandate a high FIPS level in its PKPS. On the other hand, if a bank wants a lower-cost more ergonomic solution, then the bank may allow for a lower FIPS level or even leave the FIPS level unspecified. The model element indicates the model of the credential media. For example, the model may specify a vendor and model of a particular required secure USB token.

[6]National Institute of Standards and Technologies


```
<xs:element name="CredentialAttributes">
        <xs:complexType>
                <xs:all>
                        <xs:element name="type">
                                <xs:complexType>
                                        <xs:choice>
                                                <xs:element name=”uncertified”/>
                                                <xs:element name="FIPS-140-2-level-1"/>
                                                <xs:element name="FIPS-140-2-level-2"/>
                                                <xs:element name="FIPS-140-2-level-3"/>
                                                <xs:element name="FIPS-140-2-level-4"/>
                                        </xs:choice>
                                </xs:complexType>
                        </xs:element>
                        <xs:element name="model" type="xs:string"/>
                </xs:all>
        </xs:complexType>
</xs:element>
```

Figure 19: XML schema for credential media

```
<xs:element name="SignatureMethod" type="ns1:WSPolicy"/>
        <xs:element name="SignatureMethodURI">
                <xs:complexType>
                        <xs:attribute name="URI" type="xs:anyURI"/>
                </xs:complexType>
        </xs:element>
```

Figure 20: XML schema for signature methods

Signature Method The Signature Method element specifies the expected signature format for digital signatures. The signature method specifies standards used within the financial services industry—PKCS#1 [13], PGP [6], and PSTP [4]—and a placeholder for alternative formats (i.e., “other”). For example, if a PKPS specifies PKCS#1 as the signature format and the bank receives a signature that does not comply with PKCS#1, then the bank should reject the signature immediately.

Providers The Providers section itemizes the authority requirements, as illustrated in Figure 21. When the PKPS references a provider, then the PKPS indicates an agreement between the corporation and the bank to trust a designated party for a particular purpose. The Providers section references a particular entity through a Name, DistinguishedName, and Certificate using schemas borrowed from XMLDSIG. In lieu of a Distinguished Name, the “Self” flag may be used to indicate a self-signed certificate. In all cases, the Provider section contains enough information to uniquely identify a particular entity coupled with its purpose. The PKPS recognizes three purposes:

- *CRL, OCSP:* If the revocation model is either receiver validation or sender validation with evidence, then the corporation and bank must mutually agree upon either a CRL or OCSP provider. In the case of receiver validation, the bank agrees to validate against the CRL or OCSP provider. In the case of sender validation with evidence, the signer agrees to provide evidence from the appropriate provider.
- *Timestamp:* The agreement between a particular corporation and its bank may require


```
<xs:element name="Providers" type="ns1:WSPolicy"/>
<xs:element name="CRLProvider" type="ns1:ProviderType"/>
<xs:element name="OCSPProvider" type="ns1:ProviderType"/>
<xs:element name="TimestampProvider" type="ns1:ProviderType"/>
<xs:element name="CredentialProvider" type="ns1:ProviderType"/>
<xs:complexType name="ProviderType">
        <xs:choice>
                <xs:element ref="ns1:ProviderRef"/>
                <xs:sequence>
                        <xs:element name="Name" type="xs:string"/>
                        <xs:element name="DistinguishedName" type="xs:string"/>
                        <xs:element name="CertificateData">
                                <xs:complexType>
                                        <xs:choice>
                                                <xs:element ref="ds:X509Data"/>
                                                <xs:element ref="ds:PGPData"/>
                                                <xs:element ref="ds:SPKIData"/>
                                                <xs:element name="Self"/>
                                        </xs:choice>
                                </xs:complexType>
                        </xs:element>
                </xs:sequence>
        </xs:choice>
</xs:complexType>
<xs:element name="ProviderRef" type="xs:string"/>
```

Figure 21: XML schema for the providers section

the signer to submit a signed timestamp as the claimed signing time. The timestamp provider specifies the timestamp authority upon which they mutually agree. The corporation and the bank have the flexibility to mutually agree upon a PKPS with no timestamp provider, a trusted third-party timestamp provider, or a corporate timestamper.

- *Credential:* The Credential Provider specifies the authority that issues the credential and attests to its FIPS level. The corporation and the bank have the flexibility to mutually agree upon a PKPS that allows self-signed credentials, corporate-signed credentials, or third-party signed credentials.

4 Formal Analysis with Examples

In this section, we provide a formal comparison between the PKI and PKM trust models. We use an access-control logic to highlight the underlying trust assumptions and the operations required to validate the transaction-signing keys in both the PKI and PKM models. Subsection 4.1 serves as a brief primer on the access-control logic, as it relates to our analysis. We introduce a small scenario in Subsection 4.2 that serves as the basis for our analysis. Subsection 4.3 provides a high-level comparison of PKI and the four PKM revocation models. Subsections 4.4 and 4.5 formally express the PKI and PKM models with respect to this scenario.

4.1 Access-Control Logic

To reason formally about the PKI and PKM models, we use the access-control logic described in [9] and previously used to reason about retail payment systems [8]. The syntax, Kripke-

Principal Expressions	$P ::= A \ / \ P\&Q \ / \ P \mid Q$
Access-control statements	$\varphi ::= p \ / \ \neg\varphi \ / \ \varphi_1 \wedge \varphi_2 \ / \ \varphi_1 \vee \varphi_2 \ / \ \varphi_1 \supset \varphi_2 \ / \ \varphi_1 \equiv \varphi_2 \ /$ $P \Rightarrow Q \ / \ P \ \mathsf{says} \ \varphi \ / \ P \ \mathsf{controls} \ \varphi \ / \ P \ \mathsf{reps} \ Q \ \mathsf{on} \ \varphi$
P, Q	Collection of principal expressions
A	Countable set of principal names
$P\&Q$	An abstract principal making exactly those statements made by both P and Q
$P \mid Q$	An abstract principal corresponding to principal P quoting principal Q
$P \Rightarrow Q$	P speaks for Q: informally, every statement made by P can be viewed as a statement from Q.
P controls ϕ	P is a trusted authority on statement ϕ (abbreviation for $(P \ \mathsf{says} \ \phi) \supset \phi$)
P reps Q on ϕ	P is Q's trusted delegate on ϕ (abbreviation for $(P \ \mathsf{says} \ (Q \ \mathsf{says} \ \phi)) \supset Q \ \mathsf{says} \ \phi$)

Figure 22: Syntax and informal semantics of access-control logic

$$
\begin{aligned}
\mathcal{E}_{\mathcal{M}}[\![p]\!] &= I(p) \\
\mathcal{E}_{\mathcal{M}}[\![\neg\varphi]\!] &= W - \mathcal{E}_{\mathcal{M}}[\![\varphi]\!] \\
\mathcal{E}_{\mathcal{M}}[\![\varphi_1 \wedge \varphi_2]\!] &= \mathcal{E}_{\mathcal{M}}[\![\varphi_1]\!] \cap \mathcal{E}_{\mathcal{M}}[\![\varphi_2]\!] \\
\mathcal{E}_{\mathcal{M}}[\![\varphi_1 \vee \varphi_2]\!] &= \mathcal{E}_{\mathcal{M}}[\![\varphi_1]\!] \cup \mathcal{E}_{\mathcal{M}}[\![\varphi_2]\!] \\
\mathcal{E}_{\mathcal{M}}[\![\varphi_1 \supset \varphi_2]\!] &= (W - \mathcal{E}_{\mathcal{M}}[\![\varphi_1]\!]) \cup \mathcal{E}_{\mathcal{M}}[\![\varphi_2]\!] \\
\mathcal{E}_{\mathcal{M}}[\![\varphi_1 \equiv \varphi_2]\!] &= \mathcal{E}_{\mathcal{M}}[\![\varphi_1 \supset \varphi_2]\!] \cap \mathcal{E}_{\mathcal{M}}[\![\varphi_2 \supset \varphi_1]\!] \\
\mathcal{E}_{\mathcal{M}}[\![P \Rightarrow Q]\!] &= \begin{cases} W, & \text{if } J(Q) \subseteq J(P) \\ \emptyset, & \text{otherwise} \end{cases} \\
\mathcal{E}_{\mathcal{M}}[\![P \ \mathsf{says} \ \varphi]\!] &= \{w | J(P)(w) \subseteq \mathcal{E}_{\mathcal{M}}[\![\varphi]\!]\} \\
\mathcal{E}_{\mathcal{M}}[\![P \ \mathsf{controls} \ \varphi]\!] &= \mathcal{E}_{\mathcal{M}}[\![(P \ \mathsf{says} \ \varphi) \supset \varphi]\!] \\
\mathcal{E}_{\mathcal{M}}[\![P \ \mathsf{reps} \ Q \ \mathsf{on} \ \varphi]\!] &= \mathcal{E}_{\mathcal{M}}[\![P \mid Q \ \mathsf{says} \ \varphi \supset Q \ \mathsf{says} \ \varphi]\!]
\end{aligned}
$$

Figure 23: Evaluation semantics, with $\mathcal{M} = \langle W, I, J \rangle$

style semantics, and core inference rules appear in Figures 22, 23, and 24, respectively. The inference rules are sound with respect to the semantics. The derived inference rules that we use in this paper appear in Figure 25. With these definitions in hand, we describe how to use the logic to express important concepts in our scenario, such as statements, certificates, jurisdictions, and delegation.

Statements and Certificates Principals make statements, including requests; such statements are expressed using the says operator. For example, if Alice wants to issue a payment transaction denoted by Φ_T, then Alice's request is stated as

$$\textit{Alice} \ \mathsf{says} \ \Phi_T.$$

A certificate is a signed statement. For example, in a PKI, a certificate authority signs a statement associating a cryptographic key with a principal by associating a distinguished name and a public key under the auspices of the certificate authority's signature. The receiver of the certificate in a PKI must ascertain whether the public key contained in the

$$\textit{Taut}\quad \frac{}{\varphi}\quad \text{if } \varphi \text{ is an instance of a prop-logic tautology}$$

$$\textit{Modus Ponens}\quad \frac{\varphi \qquad \varphi \supset \varphi'}{\varphi'} \qquad \textit{Says}\quad \frac{\varphi}{P \text{ says } \varphi}$$

$$\textit{MP Says}\quad \frac{}{(P \text{ says } (\varphi \supset \varphi')) \supset (P \text{ says } \varphi \supset P \text{ says } \varphi')}$$

$$\textit{Speaks For}\quad \frac{}{P \Rightarrow Q \supset (P \text{ says } \varphi \supset Q \text{ says } \varphi)}$$

$$\textit{Quoting}\quad \frac{}{P \mid Q \text{ says } \varphi \equiv P \text{ says } Q \text{ says } \varphi}$$

$$\textit{\&Says}\quad \frac{}{P \& Q \text{ says } \varphi \equiv P \text{ says } \varphi \wedge Q \text{ says } \varphi}$$

$$\textit{Idempotency of} \Rightarrow\quad \frac{}{P \Rightarrow P} \qquad \textit{Monotonicity of} \mid\quad \frac{P' \Rightarrow P \quad Q' \Rightarrow Q}{P' \mid Q' \Rightarrow P \mid Q}$$

$$\textit{Associativity of} \mid\quad \frac{P \mid (Q \mid R) \text{ says } \varphi}{(P \mid Q) \mid R \text{ says } \varphi}$$

$$P \text{ controls } \varphi \quad \stackrel{\text{def}}{=} \quad (P \text{ says } \varphi) \supset \varphi$$

$$P \text{ reps } Q \text{ on } \varphi \stackrel{\text{def}}{=} P \mid Q \text{ says } \varphi \supset Q \text{ says } \varphi$$

Figure 24: Core Inference Rules

$$\textit{Quoting (1)}\quad \frac{P \mid Q \text{ says } \varphi}{P \text{ says } Q \text{ says } \varphi} \qquad \textit{Quoting (2)}\quad \frac{P \text{ says } Q \text{ says } \varphi}{P \mid Q \text{ says } \varphi}$$

$$\textit{Controls}\quad \frac{P \text{ controls } \varphi \quad P \text{ says } \varphi}{\varphi} \qquad \textit{Derived Speaks For}\quad \frac{P \Rightarrow Q \quad P \text{ says } \varphi}{Q \text{ says } \varphi}$$

$$\textit{Reps}\quad \frac{Q \text{ controls } \varphi \quad P \text{ reps } Q \text{ on } \varphi \quad P \mid Q \text{ says } \varphi}{\varphi}$$

$$\textit{Rep Says}\quad \frac{P \text{ reps } Q \text{ on } \varphi \quad P \mid Q \text{ says } \varphi}{Q \text{ says } \varphi}$$

Figure 25: Derived Rules Used in this Paper

certificate is currently active. For example, suppose that Alice obtains a certificate from *CA*. Alice's key certificate can be formally expressed as

$$CA \text{ says } (\langle K_A, Active\rangle \supset K_A \Rightarrow Alice),$$

where $\langle K_A, Active\rangle$ is a proposition that reflects the status of the key K_A. Informally, CA says if K_A is active, then K_A speaks for Alice; and the receiver requires an extra step beyond certificate validation to determine whether K_A is active.

Authority and Jurisdiction Jurisdiction statements identify who or what has authority, specific privileges, powers, or rights. In the logic, jurisdiction statements are typically expressed via the controls operator. For example, if a bank believes in the authority of *CA* to issue a certificate, then we write

$$CA \text{ controls } (\langle K_A, Active\rangle \supset K_A \Rightarrow Alice).$$

If CA has authority to issue a certificate and subsequently issues that certificate, then the *Controls* inference rule in Figure 25 allows us to infer the validity of the certificate:

$$\frac{\begin{array}{c} CA \text{ controls } (\langle K_A, Active\rangle \supset K_A \Rightarrow Alice) \\ CA \text{ says } (\langle K_A, Active\rangle \supset K_A \Rightarrow Alice) \end{array}}{\langle K_A, Active\rangle \supset K_A \Rightarrow Alice.}$$

Furthermore, if the bank can verify that $\langle K_A, Active\rangle$ is true (for example, by receiving an OCSP response), then the bank can conclude that $K_A \Rightarrow Alice$ using the following derived inference rule:

$$\frac{\langle K_A, Active\rangle \supset K_A \Rightarrow Alice \qquad \langle K_A, Active\rangle}{K_A \Rightarrow Alice.}$$

Proxies and delegates In an electronic transaction, the cryptographic key used to sign the transaction serves as a proxy for the principal who submits the transaction to the bank. For example, suppose that Alice uses her key K_A to issue a transaction to the bank. If the bank trusts that the key K_A belongs to Alice (i.e., $K_A \Rightarrow Alice$), then the bank can attribute all statements made using K_A to Alice. Using the *Derived Speaks For* rule in Figure 25, the bank can deduce that the transaction signed by K_A came from Alice:

$$\frac{K_A \Rightarrow Alice \qquad K_A \text{ says } \Phi_T}{Alice \text{ says } \Phi_T.}$$

In some situations, a principal may be trusted only on specific statements. This notion of constrained delegation is described using the reps operator. For example, if K_A is trusted to be Alice's delegate on the statement Φ_T, then we can write

$$K_A \text{ reps } Alice \text{ on } \Phi_T.$$

The semantics of reps ensures that, if we recognize K_A as Alice's delegate, then we are in effect saying that K_A is trusted on Alice to issue transaction Φ_T. If K_A says Alice says Φ_T, then we write

$$K_A \mid Alice \text{ says } \Phi_T$$

We can then use the *Rep Says* rule from Figure 25 to conclude that Alice has made the request:

$$\frac{\begin{array}{c} K_A \text{ reps } Alice \text{ on } \Phi_T \\ K_A \mid Alice \text{ says } \Phi_T \end{array}}{Alice \text{ says } \Phi_T.}$$

4.2 Sample Scenario

To illustrate the similarities and differences among PKI and the PKM revocation models, we introduce a sample scenario where a corporation C sends a transaction Φ_T to a financial institution F. Alice, Bob, and Doug are employees of C and are assigned public keys denoted by K_A, K_B, and K_D, respectively. Alice, Bob, and Doug hold respective roles R_1, R_2, and R_3 that are assigned by C and recognized by F. A benevolent third-party certificate authority

Function	Notation
Corporation	C
Financial Institution	F
Alice, employee of C	A
Bob, employee of C	B
Doug, employee of C	D
Alice's role	R_1
Bob's role	R_2
Doug's role	R_3
Alice's key	K_A
Bob's key	K_B
Doug's key	K_D
Certificate authority	CA
Mutually agreed PKPS	Φ_{PKPS}

Figure 26: Notation

```
<SignaturePolicy>
 <Policy>
  <ExactlyOne>
   <Roles>
    <Role> R1 </Role>
    <Role> R2 </Role>
    <Role> R3 </Role>
   <Roles>
  </ExactlyOne>
 </Policy>
</SignaturePolicy
```

Figure 27: Signature Policy

CA is also used in some instances. In the case of PKM, C and F agree on a PKPS denoted Φ_{PKPS}. Figure 26 summarizes the notation.

C and F agree to impose a signature policy that requires F to reject any incoming transaction that does not have signatures from three distinct people acting in the roles R_1, R_2, and R_3, respectively. In the PKM model, C and F can enforce such signature policies using the signature policy section of the PKPS (see Figure 27). In contrast, when using PKI, C and F have to use other proprietary methods for enforcing the policy.

Transaction Request Alice, Bob, and Doug sign a transaction Φ_T using their respective keys K_A, K_B, and K_D and asserting their respective roles R_1, R_2, and R_3. The transaction request can be formally stated as follows:

$$(K_A \mid R_1)\&(K_B \mid R_2)\&(K_D \mid R_3) \text{ says } \Phi_T.$$

Signed PKPS Additionally, Alice asserts her role R_1 to sign a PKPS document that she includes in the transaction. When F receives the transaction and matches it to Φ_{PKPS} in its repository, the signed PKPS can be formally stated as follows:

$$(K_A \mid R_1) \text{ says } \Phi_{PKPS}.$$

The financial institution will act on the transaction request if it is able to conclude Φ_T. In the remainder of this section, we describe the validation processes under both PKI and PKM, showing how F concludes Φ_T.

4.3 Comparison of PKI and PKM Models

From the bank's perspective, the answers to the following three questions characterize the underlying trust assumptions and operations of both the PKI and PKM key-validation methods:

1. *Who decides when a certificate should be valid?*

	PKI	PKM-RV	PKM-SVE	PKM-SVNE	PKM-SVCS
Who decides when a certificate should be valid?	C	C	C	C	C
Who has authority to quote C for status?	CA	CA	CA	N/A	N/A
Who issues the credential ?	CA	CA	CA	C	C

PKI: Public key infrastructure
PKM: Partner key management
RV: Receiver validation
SVE: Sender validation with evidence
SVNE: Sender validation without evidence
SVCS: Sender validation with cosign
C: Corporation
N/A: Not applicable
CA: Certificate authority

Table 1: Comparison of PKI and PKM from a bank's perspective

An aspect of commonality between the PKI and the PKM is the authority who determines a key's current validity. In wholesale banking, a corporation's authorized administrators or the key owner determine the key's current status. If the corporation uses a certificate authority such as *CA*, then the corporation's administrators or the key owner instruct *CA* on the key's current status.[7]

2. *Who has authority to quote the corporation on current status of certificate?*

   Both the PKI and PKM need to understand the current validity of a certificate during the validation sequence. However, they may differ in their technical means of discovery. Customarily, PKI employs the receiver-validation model; however, no technical prohibition stops the PKI from adopting sender-validation-with-evidence. In neither of these models do the bank and the corporation directly communicate to discover the certificate's current status. Instead, the corporation communicates the certificate's current status to the bank using the certificate authority. In contrast, PKM's sender-validation-without-evidence and sender-validation-with-cosign models do not use a certificate authority because the corporation directly transmits current status of its certificates to each of the banks without relying upon a middleman.

3. *Who issues the credential?*

   The concept of issuing credentials is very important in PKI, because of the need to secure a trusted distinguished name. If the credential issuer is not trustworthy, then the issuer could potentially provide a certificate marked with a particular distinguished name to the wrong party. In contrast, PKM ignores the distinguished name; and in some models certificate issuance and revocation have a relationship.

The next two sections illustrate the trust assumptions and operations in each of the four trust models in detail.

[7]For expository purposes, in our subsequent analysis, we focus on the corporation's authority and ignore the user's authority. Accounting for the user's authority requires only small changes to the relevant statements of jurisdiction.

4.4 Public Key Infrastructure

There are three core trust assumptions in the PKI model, plus a reliance on the receipt of an appropriate certificate:

1. The CA is the authority for issuing the credentials to all the employees of C:

$$CA \text{ controls } (\langle K_A, \mathit{Active}\rangle \supset K_A \Rightarrow \mathit{Alice}).$$

2. The corporation is the authority for determining the current status of the keys:

$$C \text{ controls } \langle K_A, \mathit{Active}\rangle.$$

3. CA is a delegate of the corporation C for communicating the status of the keys to the financial institution:

$$CA \text{ reps } C \text{ on } \langle K_A, \mathit{Active}\rangle.$$

   Typically, the CA maintains an OCSP responder or a CRL to communicate the status of the keys to relying parties such as F. When the CA relays a statement from C that K_A is active, we write

$$CA \mid C \text{ says } \langle K_A, \mathit{Active}\rangle$$

4. The CA-issued certificate asserts that, if the key is active, then the key K_A is associated with Alice:

$$CA \text{ says } (\langle K_A, \mathit{Active}\rangle \supset K_A \Rightarrow \mathit{Alice}).$$

The inference rule for key validation (i.e., concluding $K_A \Rightarrow \mathit{Alice}$) under the PKI model can be formally stated as follows:

$$\frac{\begin{array}{c} CA \text{ controls } (\langle K_A, \mathit{Active}\rangle \supset K_A \Rightarrow \mathit{Alice}) \\ C \text{ controls } \langle K_A, \mathit{Active}\rangle \\ CA \text{ reps } C \text{ on } \langle K_A, \mathit{Active}\rangle \\ CA \text{ says } (\langle K_A, \mathit{Active}\rangle \supset K_A \Rightarrow \mathit{Alice}) \\ CA \mid C \text{ says } \langle K_A, \mathit{Active}\rangle \end{array}}{K_A \Rightarrow \mathit{Alice}.}$$

This rule states that, for the financial institution to conclude that the key K_A speaks for Alice, it must rely on the four trust assumptions and also receive a message from CA on C's behalf.

4.5 Partner Key Management

For illustrative purposes, we describe the PKM validation process under each of the four revocation models. All four revocation models under PKM share variations of the following two trust assumptions:

1. Credentials are issued either by C or CA, depending upon the revocation model. The authority for issuing credentials to C's employees can be expressed in the following general form:

$$X \text{ controls } (\langle K_A, \mathit{Active}\rangle \supset K_A \Rightarrow \mathit{Alice}).$$


```
<revocation>
 <policy>
  <receiver-validation>
   <OCSPProvider>CA</OCSPProvider>
  </receiver-validation>
 </policy>
</revocation>
```

$$\frac{\begin{array}{c} CA \text{ controls } (\langle K_A, Active\rangle \supset K_A \Rightarrow Alice) \\ C \text{ controls } \langle K_A, Active\rangle \\ CA \text{ reps } C \text{ on } \langle K_A, Active\rangle \\ CA \text{ says } (\langle K_A, Active\rangle \supset K_A \Rightarrow Alice) \\ CA \mid C \text{ says } \langle K_A, Active\rangle \end{array}}{K_A \Rightarrow Alice}$$

Figure 28: PKPS section and inference rule for receiver validation

In the receiver-validation and sender-validation with-evidence models as follows, X is instantiated with CA; in the sender-validation without-evidence and sender-validation with-cosign models, X is instantiated with C.

2. In all cases, the corporation C is the authority for determining the key's status:

$$C \text{ controls } \langle K_A, Active\rangle.$$

In addition to these two trust assumptions, the receiver-validation and sender-validation-with-evidence models require an additional trust assumption:

$$CA \text{ reps } C \text{ on } \langle K_A, Active\rangle.$$

That is, the financial institution must recognize CA as a trusted delegate of C with regards to whether the key K_A is active.

The four revocation models vary in how C communicates the status of the keys to F. In the remainder of this section, we describe the details of each of the four revocation models for our scenario.

Receiver Validation Suppose that CA uses the OCSP protocol to communicate the status of the keys. Figure 28 shows the PKPS revocation section and the inference rule for this model. The trust assumptions underlying PKI and the PKM receiver-validation model are equivalent. In both models, CA issues the keys, and the corporation is the authority on the key's current status and communicates the status using CA.

Sender Validation with Evidence Figure 29 shows the PKPS section and inference rule under the sender-validation-with-evidence model. The formalization of this model is identical to that for the receiver-validation model, because the difference between the two models is purely mechanical and not logical. Using receiver validation, the bank obtains the information directly from the OCSP responder. In contrast, in the sender-validation-with-evidence model, the bank receives the same information indirectly through the corporation. Because the OCSP responder use digital signatures to sign their statements, the corporation cannot forge these statements.

Sender Validation without Evidence The sender-validation-without-evidence model removes the benevolent third party from the interaction (and hence CA does not show up in the inference rule). Figure 30 contains the PKPS section and inference rule for this model.


```
<revocation>
 <policy>
  <sender-validation-with-evidence>
   <OCSPProvider>CA</OCSPProvider>
  </sender-validation-with-evidence>
 </policy>
</revocation>
```

$$\frac{\begin{array}{c} CA \text{ controls } (\langle K_A, Active\rangle \supset K_A \Rightarrow Alice) \\ C \text{ controls } \langle K_A, Active\rangle \\ CA \text{ reps } C \text{ on } \langle K_A, Active\rangle \\ CA \text{ says } (\langle K_A, Active\rangle \supset K_A \Rightarrow Alice) \\ CA \mid C \text{ says } \langle K_A, Active\rangle \end{array}}{K_A \Rightarrow Alice}$$

Figure 29: PKPS section and inference rule for sender validation with evidence

```
<revocation>
 <policy>
  <sender-validation-
      without-evidence/>
 </policy>
</revocation>
```

$$\frac{\begin{array}{c} C \text{ controls } (\langle K_A, Active\rangle \supset K_A \Rightarrow Alice) \\ C \text{ controls } \langle K_A, Active\rangle \\ C \text{ says } (\langle K_A, Active\rangle \supset K_A \Rightarrow Alice) \\ C \text{ says } \langle K_A, Active\rangle \end{array}}{K_A \Rightarrow Alice}$$

Figure 30: PKPS section and inference rule for sender validation without evidence

Accordingly, the trust assumptions are similar to the those for the previous models, except that there is no proxy relationship in this setting. As opposed to indirectly informing the bank using a third party, C communicates the status of the keys to F directly using the PKM protocol.

Sender Validation with Cosign Under this model, whenever Alice signs a transaction, C is also expected to sign Alice's statement. Figure 31 contains the PKPS section and inference rule for this model. In addition to the standard PKM trust assumptions, the following statements characterize this model:

1. In reference to the operating rules mutually agreed offline between the corporation and the bank, whenever C cosigns statement, C states that Alice's key is currently active without posing additional assertions concerning the validity of Φ_T:

$$C \text{ says } K_A \text{ says } \Phi_T \supset C \text{ says } \langle K_A, Active\rangle.$$

2. F knows the public key of C, and therefore can attribute any statement signed by K_C to C:

$$K_C \Rightarrow C.$$

3. In a transaction, when C cosigns the transaction request using K_C, we write:

$$K_C \text{ says } K_A \text{ says } \Phi_T.$$


```
<revocation>
 <policy>
  <sender-validation-
         with-co-sign/>
 </policy>
</revocation>
```

$$\frac{\begin{array}{c} C \text{ controls } (\langle K_A, \mathit{Active}\rangle \supset K_A \Rightarrow \mathit{Alice}) \\ C \text{ controls } \langle K_A, \mathit{Active}\rangle \\ C \text{ says } (\langle K_A, \mathit{Active}\rangle \supset K_A \Rightarrow \mathit{Alice}) \\ C \text{ says } K_A \text{ says } \Phi_T \supset C \text{ says } \langle K_A, \mathit{Active}\rangle \\ K_C \Rightarrow C \quad K_C \text{ says } K_A \text{ says } \Phi_T \end{array}}{K_A \Rightarrow \mathit{Alice}}$$

Figure 31: PKPS section and inference rule for sender validation with cosign

4.6 Transaction Model

The inference rule for the final step of transaction validation can be stated as follows:

$$\frac{\begin{array}{c} (K_A \mid R_1)\&(K_B \mid R_2)\&(K_D \mid R_3) \text{ says } \Phi_T \\ K_A \Rightarrow \mathit{Alice} \quad K_B \Rightarrow \mathit{Bob} \quad K_D \Rightarrow \mathit{Doug} \\ \mathit{Alice} \text{ reps } R_1 \text{ on } \Phi_T \quad \mathit{Bob} \text{ reps } R_2 \text{ on } \Phi_T \quad \mathit{Doug} \text{ reps } R_3 \text{ on } \Phi_T \\ C \text{ controls } R_1\&R_2\&R_3 \text{ controls } \Phi_T \\ C \text{ says } R_1\&R_2\&R_3 \text{ controls } \Phi_T \end{array}}{\Phi_T}.$$

The first line corresponds to the initial transaction request. The second line is the result of validating keys under either the PKI or PKM model. The third line represents the role assignment. The method of assigning roles to individuals is outside the scope of PKM/PKI, but within the scope of each bank's authorization method. The fourth and fifth lines assert that the corporation has jurisdiction over which roles are necessary and sufficient for directing transactions. All these statement are necessary for F to conclude Φ_T.

In conclusion, the PKI and all four revocation models yield the same result: sound reasoning permits F to conclude Φ_T and thereby safely process the transaction. Furthermore, upon close inspection of the logical steps required in the demonstration, one can see that the difference in security between PKI and the four revocation models is a mere technicality: the PKI and some of PKM models require the CA as a middleman, and other PKM models do not require a middleman.

5 Related Technologies

We propose PKM for providing credential interoperability in wholesale banking and possibly other industries. A related technology is the Security Assertion Markup Language [7] (SAML), which is a standard for providing identity interoperability for a collection of web-based services. SAML enables services such as single-sign on (SSO). SAML and other interoperable SSO methods do not offer signatures, which make them ill-suited for satisfying the needs of contract law.

A PKPS is a set of constraints upon credentials. We chose WS-Policy [18] for our implementation, because it is better suited to express these constraints than the alternatives of WSPL [2], XACML [16], X.509 extensions, and P3P [11]. WS-Policy is a W3C standard for specifying web-service policies for security, quality of service, messaging, and other non-functional requirements. WSPL has similar abilities, but it is not an accepted W3C standard. XACML is a declarative language for specifying access-control policies. Although the PKPS

constraints may be crafted like access-control policies, a language designed to express constraints and capabilities such as WS-Policy is a better match. X.509 extensions use ASN.1 notation for specifying constraints on certificates, but they do not enjoy universal acceptance. Moreover, XML provides a more modern format that is more readable than ASN.1. P3P [11] is language designed for expressing privacy preferences and is not suited for expressing PKPS constraints.

6 Conclusion

The primary purpose of Partner Key Management (PKM) is to provide a credential-management mechanism that is well suited for business practices. The justification for PKM resides within contract law, which lies at the core of business. With the possible exception of national governments, business normally recognizes parties who operate under the legal obligations of their contracts. This paper uses formal methods to compare the relative security strength of PKI and the four revocation models and concludes that all four models provide equal security when regarded from the perspective of contract law. In fact, the derived inference rules for all four model possess inherent structural similarities, which illustrate that the CA merely plays the role of a middleman. PKM has the facility to remove the CA's middleman requirement without sacrificing security. Each business domain should look to common business and technical practices when choosing the optimal revocation model. Furthermore, no one should expect all industries to leverage a single optimized model.

This paper focuses on wholesale banking by highlighting the business requirements for interoperability. Wholesale banking may participate in a banking community by leveraging interoperable credentials; however, wholesale banking does not require or need interoperable identities. Furthermore, wholesale banking must afford significant autonomy to each bank that participates in the community. The receiver-validation model is an ill-fit for wholesale banking, because it requires the banks to establish complex interbank identity management without benefiting from a significant business advantage. Furthermore, the receiver-validation model mandates that banks diminish their autonomy by requiring them to undergo further international standardization and possibly harmonization of national government regulatory practices.

The various sender-validation models better reflect wholesale business practices. The path from current business practices toward interoperable credentials under the auspices of sender validation is relatively straight-forward. Banks may continue to leverage most of their existing practices, and national regulatory agencies do not need to change.

A move to a sender-validation model within the wholesale banking industry would provide a welcome opportunity for a technology refresh. The X.509 standard is now twenty-two years old, and the industry needs to update based on new technology from two perspectives. First, by definition, interoperability requires communication within a community. Thus, self-describing XML provides a better technology choice than out-of-date ASN.1 syntax. The Partner Key Policy Statement (PKPS) is the XML document that addresses this need. Second, web services technology did not exist when the industry invented the concept of a Certificate Authority. Today, the industry is in the process of rapidly adopting web services. In wholesale banking, corporations should upgrade their infrastructure to use web services to implement the PKM protocol, thereby communicating with each of its banks di-

rectly and discarding any requirement for a benevolent certificate authority operating as a middleman.

While the PKI falls short in meeting all of wholesale banking's high level requirements, PKM's receiver validation model is an excellent match.

- By implementing a direct interaction between the corporation and each bank, the PKM protocol circumvents the liability issue.
- By removing the middleman role played by the certificate authority, the bank positions itself with the ability to delegate credential administration efficiently to its corporate customers.
- By observing the receiver validation model, the banks participate in an agile marketplace because they do not need to connect to an array of CRLs or OCSP providers.
- By permitting each bank its choice of PKPS, the industry may immediately adopt PKM without waiting for global banking standardization.

Given the goal of credential interoperability, wholesale banking has two choices. Wholesale banking can either upgrade its technological infrastructure by implementing the PKM protocol through web services, or wholesale banking may modify its business practices and regulations by leveraging bridged PKIs. The barrier of entry for the first choice is dramatically lower than the second choice because PKM does not require changing business practices. No significant barriers exist today that prevent wholesale banking from initiating its move to PKM and ultimately achieving the goal of credential interoperability.

References

[1] C. Adams, S. Farrell, T. Kause, and T. Mononen. Internet X.509 Public Key Infrastructure Certificate Management Protocol (CMP). RFC 4210 (Proposed Standard), Sept. 2005.

[2] A. H. Anderson. An introduction to the web services policy language (wspl). In *Proceedings of the Fifth IEEE International Workshop on Policies for Distributed Systems and Network*, pages 189–, Washington, DC, USA, 2004. IEEE Computer Society.

[3] M. Bartel, J. Boyer, B. Fox, B. LaMacchia, and E. Simon. XML Signature Syntax and Processing (Second Edition). Technical report, IETF/W3C XML Signature Working Group, 2008.

[4] G. Benson. Portable security transaction protocol. *Comput. Netw.*, 51:751–766, February 2007.

[5] T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, and F. Yergeau. Extensible Markup Language (XML) V1.0. Technical report, W3C XML core working group, Nov 2008.

[6] J. Callas, L. Donnerhacke, H. Finney, D. Shaw, and R. Thayer. OpenPGP Message Format. RFC 4880 (Proposed Standard), Nov. 2007. Updated by RFC 5581.

[7] S. Cantor, J. Kemp, R. Philpott, and E. Maler. Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0. Technical report, OASIS Security services technical committee, 2005.

[8] S.-K. Chin and S. Older. Reasoning about delegation and account access in retail payment systems. In *MMM-ACNS*, 2007.

[9] S.-K. Chin and S. Older. *Access Control, Security, and Trust : A Logical Approach*. Chapman and Hall/CRC, 1 edition, July 2011.

[10] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 5280 (Proposed Standard), May 2008.

[11] L. Cranor, M. Langheinrich, M. Marchiori, M. Presler-Marshall, and J. Reagle. The Platform for Privacy Preferences 1.0 (P3P1.0) Specification. Technical report, W3C Technology & Society Domain, April 2002.

[12] D. L. Evans, P. J. Bond, and A. L. Bement. Security requirements for cryptographic modules. Technical Report FIPS PUB 140-2, National Institute of Standards and Technology - Information Technology Laboratory, Gaithersburg, MD 20899-8900, May 2001.

[13] J. Jonsson and B. Kaliski. Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1. RFC 3447 (Informational), Feb. 2003.

[14] J. Lim. Korea national pki status and directions for market promotion, 2009.

[15] Y. Miyakawa, T. Kurokawa, A. Yamamura, and Y. Matsumoto. Current status of japanese government pki systems. In *Proceedings of the 5th European PKI workshop on Public Key Infrastructure: Theory and Practice*, EuroPKI '08, pages 104–117, Berlin, Heidelberg, 2008. Springer-Verlag.

[16] T. Moses. eXtensible Access Control Markup Language (XACML) V2.0. Technical report, OASIS Access Control TC, Feb 2005.

[17] M. Myers and H. Tschofenig. Online Certificate Status Protocol (OCSP) Extensions to IKEv2. RFC 4806 (Proposed Standard), Feb. 2007.

[18] A. S. Vedamuthu, D. Orchard, F. Hirsch, M. Hondo, P. Yendluri, T. Boubez, and U. Yalcinalp. Web Services Policy 1.5 - Framework. Technical report, W3C Web Services Policy Working Group, September 2007.

