國 立 交 通 大 學

電機學院與資訊學院 資訊學程

碩士論文

運用 XML 建立保護式 DSA 代理簽章方法之研究

Study of Proxy-protected DSA Scheme Based on XML

研 究 生：吳俊欽

指導教授：葉義雄 教授

中 華 民 國 九 十 五 年 五 月

運用 XML 建立保護式 DSA 代理簽章方法之研究

Study of Proxy-protected DSA Scheme Based on XML

研 究 生：吳俊欽　　Student：Chun-Chin Wu

指導教授：葉義雄　　Advisor：Dr. Yi-Shiung Yeh

國 立 交 通 大 學

電機學院與資訊學院 資訊學程

碩 士 論 文

A Thesis
Submitted to Degree Program of Electrical Engineering and Computer Science
College of Computer Science
National Chiao Tung University
in Partial Fulfillment of the Requirements
for the Degree of
Master of Science
in
Computer Science

May 2006

Hsinchu, Taiwan, Republic of China

中 華 民 國 九 十 五 年 五 月

運用 XML 建立保護式 DSA 代理簽章方法之研究

學生:吳俊欽　　　　　　　指導教授:葉義雄博士

國立交通大學電機學院與資訊學院 資訊學程（研究所）碩士班

摘　要

現今的網際網路已經充斥了整個世界，不論是購物、繳稅、掛號、訂位等，皆脫離不了網際網路，相對地也提供了極大的便利與效率，尤其 B2B、B2C 的盛行，帶動了世界整體經濟的發展，也拉近了國際間，甚至人與人之間的距離；但是，網際網路基本上是一個不安全、無法辨識個體的環境，任何人都可以假冒、竊取或偽造網路上的資訊，如此便容易造成許多的糾紛，所以資訊安全以及身份認證就成為了推動網路交易一個重要議題。

提供一個保密且安全的網路傳輸通道是最基本且無庸置疑的，另外，如何在一筆交易間互相辨識及鑑定，也是促成網路交易的重要因素之一；數位簽章即為身份認證應運而生的一個機制，藉由單向赫序函數不可逆特性或解離散對數困難度使得簽章難以被偽造，僅能使用私密金鑰才能驗證文件為簽署者所簽署，進而達到身份認證的目的。

目前有許多種各式電子文件之格式，例如：WORD 之 DOC 檔、ACROBAT 之 PDF 檔、HTML…等，而做為世界廣泛認可的數位簽章文件格式必須具備開放、具有彈性，且有公認發行格式之特性，而 XML（eXtensible Markup Language）則具備以上的特點，非常適合運用於電子簽章與文件之結合。

所以，如果運用 XML 來實作代理簽章，不僅可在網路平台上易於推行具公信力的網路交易或電子公文，而且統一格式更可以強化文件交換之流通性，尤其可符合一般公司之運用情況，例如：董事長因公外出，總經理必須代理其職權，則必須使用代理簽章；由於目前 XML 所建立公認之簽章格式並不包含代理簽章格式，本文即針對該規範進行代理簽章之延伸，並實作文件代理簽章之做法。

Study of Proxy-protected DSA Scheme Based on XML

Student: Chun-Chin Wu **Advisor: Dr. Yi-Shiung Yeh**

Degree Program of Electrical Engineering and Computer Science

National Chiao Tung University

Abstract

The world, nowadays, has been flooded with internet. No matter where is in shopping, tax payment, registering in hospital, and reservation for seat and so on. They all can't be departed from internet. It also provides extreme convenience and efficiency relatively. Especially in the vogue by B2B or B2C, it not only pushes forward the developments of whole world economics, but also gets closer between international, even person to person. But internet is basically an unsecured and unidentified environment. Anyone can counterfeit, steal or forge the information over internet, and so that it would be easy to make many disputes. As the result, information security and identity certification will be an important issue for giving an impetus to network transactions.

There is no doubt but fundamental to provide a confidential and safe transmission channel for network. It is also, on the other hand, an important factor how to identify or authenticate each other in a transaction. Digital signature is such a mechanism coming with the tide of fashion for authentication. The signature is hard to be forged based on irreversible one-way hash function or discrete logarithm. It just only uses the private key to prove the document being signed by signer in order to achieve the purpose of authentication.

There are, at present, many kinds of e-documents such like DOC files for WORD, PDF files for Acrobat, HTML and so on. But it must have characteristics of open, flexible, public acknowledged format to be a worldwide approved digital signature format. XML (eXtensible Markup Language) would have those of characteristics, and it is very suitable for combining digital signatures with documents.

So, if proxy signature can be implemented by XML, the trusted network transactions or electronic official documents would be easily carried out. Unified format can, moreover, enhance the circulation of document exchange; it is especially suitable for applying to general companies. The president, for example, is away on official business; the general manager must act for the president. Proxy signature must be used in this case. Owing to the signature format acknowledged by XML does not support proxy signature, this thesis will extend the standard format for proxy signature and implementation.

致　　謝

能夠順利完成論文，首先最感謝我的指導教授葉義雄博士，在這三年內不論在研究及學校方面都給予我極大的指導與協助，使我在這麼短的時間能夠進入密碼學的領域，並且在我就讀研究所這一段時間裡，不僅在學習上或者是解決問題上，給我完整的指導與協助，也就是在葉老師孜孜不倦的教導下，終於完成我生平第一份著作：我的碩士論文。

再來，感謝我的父母一直很支持我完成我的學業，雖然媽媽在我還未完成學業之前已經先離我而去，但希望她在天之靈可以看到我拿到衷心期盼的碩士學位。另外感謝陳以德學長及黃定宇學長，給了我許多的支持與建議，其中最由衷感佩的是在前年已順利獲得博士學位的張明信學長，在他完成博士學位前這一段最忙碌的時刻，還非常有耐心地帶著我一步步完成我的論文，在口試前後，也給我非常多寶貴的建議，一路走來，滿心感謝。

在口試期間，謝謝蔡文能及黃世昆兩位委員老師的指正，尤其黃老師精闢的提問與見解，提供了非常寶貴的意見，也讓我的論文能更加完美；而蔡老師是我的偶像，尤其先前上過蔡老師的網路安全，真的讓我獲益良多，也使得我踏進密碼學這個領域的腳步更加穩健。

最後，我所要感謝的是我的妻子怡如及已經滿週歲的女兒昀臻，有她們在背後支持著我，尤其每次看到可愛的女兒對我微笑，使得這一切的壓力與煩惱都在一瞬間煙消雲散，她是我的開心果，也是心靈的加油站；這一切也要感謝我的老婆幫我好好照顧整個家，讓我無後顧之憂，順利地完成學業，她是我背後最大的支柱，所以，我要在此說一聲：老婆還有小臻，我愛妳們。

吳俊欽　謹誌

中 華 民 國　九 十 五 年 五 月

Contents

List of Tables

List of Figures

Chapter 1 Introduction

The modern of the computer technology are progressing quickly, so the information security and authentication plays an important role over the internet application. It basically requires a flexible and unique platform packing the confidential information in the network. Everyone can handover crucial and lawful documents conveniently. XML is one of the suitable formats for data exchange.

The XML (Extensible Markup Language) [W3C02] is a simplification from SGML (Standard Generalized Markup Language) being formulated by "World Wide Web Consortium", and became the specification of recommendation in February of 1998. There are a lot of manufacturers have adopted the XML at present, and it has been regarded as a key technology. For example: Adobe, IBM, Microsoft, Netscape, Oracle, and many important software manufacturers are using XML to develop some applications. A lot of new software, moreover, such like Internet Explorer browser, RealPlayer and so on, has already used the XML technology.

The essential success of XML is owing to some advantages. As computers join in the network, they would exchange data each other to obtain information. The data has to be converted to a sequence of stream and the mechanism is called “serialized”. The received side, on the other hand, must decode those of serialized data correctly. XML is the only one standard format serializing the data.

We know the digital signature is crucial to companies and governments because all trades, effective certificates or authorities via network need an identical demonstration. The digital signature has a non-repudiation characteristic and so the digital signature is the good way for identity authentication.

We also aware the digital signature just only signed by himself / herself. The chief, however, is on a business trip or holiday; but company is still working. If the person who is on behalf of the chief signs the document has equal effect, so that will be helpful to affairs in the company. There is no influence with having transactions with the other companies because the deal has been made by delegation. It will fill the bill to have delegation mechanism in commercial trading. Proxy signature is a good algorithm for practical requirement in business.

There are many kinds of digital signatures, for instance, DSA, RSA and so on. Proxy signature is the extension of the digital signature. The proxy signature allows a designated person, called a proxy signer, to substitute for an original signer to sign the document. The proposed proxy signature scheme is based on the discrete logarithm problem. Compared to the consecutive execution of the ordinary digital signature schemes, it has a direct form, and a verifier does not need a public key of a user other than the original signer when verification.

The main purpose of this thesis is to illustrate and implement the proxy signature based on XML. As we mention before, all transactions or commerce can't be departed from network. XML is a suitable medium for implementation. There are many advantages using XML. It can, moreover, be easily attached in the file, standalone or figure out by URI; XML is also a readable pure text too. Proxy signature is presented to printable characters so that it could be identified by person eyes and easy to know who signed the document. XML is the worldwide and all-purpose format. We can apply the proxy signature on XML in different system without modification, and just follow the XML format.

Chapter 2 Motivation

To catch up with the trends of world, and electronic trading and commercial activities in the network is more and more inseparable with our daily life. But it is an essential subject of debate how do we establish the trust system during transaction. The use of signature would be helpful to build up a safe transaction in the virtual world. The traditional signature uses the seal or some sort of pledge stamp to be the proof for the effective documents, but counterfeiters or forgers can behalf of him/her to delegate the power. There is going without saying that proxy person is maybe the forger, too. He/She signs the document under unauthorized circumstance, and will influence on a great range from an individual even to a company.

There is no doubt that we also suffered the same situation in network, so it must have an appropriate method to avoid these problems. First stuff we should consider is the signature algorithm. There are some kinds of signature standard available and deployed. DSA or RSA are typical cases, and they already had been examined and used for years. Both of them, unfortunately, have not proxy property; they just can be called the conventional signature. The mechanisms only provide the signer signs the document himself/herself without delegated capability. Many people dedicate themselves in proxy signature research.

The proxy signature can be divided two categories: proxy unprotected and protected scheme for partial delegation [$MUO_1$96, $MUO_2$96], the proxy protected scheme is a better solution although proxy unprotected one has good performance. We have more detail of discussion at latter chapter.

Many research in proxy signature, up to till now, cease moving across the application region but only improving or modifying their architectures or algorithms. This thesis dedicates to combine with two different kind of knowledge and the goal we want to achieve is to apply the proxy signature over network. That would help to construct an applicable and trusted proxy mechanism for present transactions or working procedure in network.

We have to decide what type of format is used for network after the proxy signature is confirmed. XML, we have mention in previous chapter, is the very kind of language. XML signature had been deployed for many years, but it is not all fit in with the requirement of proxy signature. There would be a lot of works to do if we changed the structure or add some new features to current system. We must try to conceal new parameters in some place in order to keep working compatibly with XML signature.

The main issue of this thesis, we have mentioned before, determines how does the proxy protected DSA scheme being applied to XML signature and compatible with current format. We also take all kind of source data applying to three kind of the way to signature packaging to determine the practicality and expansibility for our proposed algorithm.

This is the list of related knowledge about this thesis as following:

- Digital Signature Algorithm (DSA).
- Proxy signature.
- Proxy unprotected and protected scheme.
- Proxy protected DSA scheme.
- eXtensible Markup Language (XML).
- XML signature.
- XPATH and Manifest tags.

- Enveloped, enveloping and detached signature.

In following chapters, we will discuss the XML first. Secondly, we are going to describe about proxy signature. Chapter 5, 6 is determined the algorithm we proposed and how to design all features corresponding to our requirement. The end of two chapters will discuss successful efforts we have done and which parts of works needed to be enhanced in the future.

Chapter 3 XML

XML is a kind of markup language for documents containing structured information [NW98]. Structured information contains content likes words, pictures or binary data. A markup language is a mechanism to identify structures and what it stands for in a document. The XML specification defines a standard way to add markup to documents. XML is also a plentifully structured document and that could be used over the web. The HTML and SGML, the other kinds of markup language, are not practical for this thesis.

HTML comes bound with a set of semantics and does not provide arbitrary structure. It can not extend some feature for use in some purpose. Although SGML provides arbitrary structure, it is too difficult to implement just for a web browser. Full SGML systems are quite large, complex problems that justify their expense. Viewing structured documents sent over the web rarely carries such justification. But XML is not expected to completely replace SGML, it is designed to deliver structured content over the web, some of the very features it lacks to make this practical, make SGML a more satisfactory solution for the creation and long-time storage of complex documents. Simplifying SGML to XML will be, in many institutions, the standard procedure for web delivery. The figure will expound the scope of these three markup languages we discuss above:

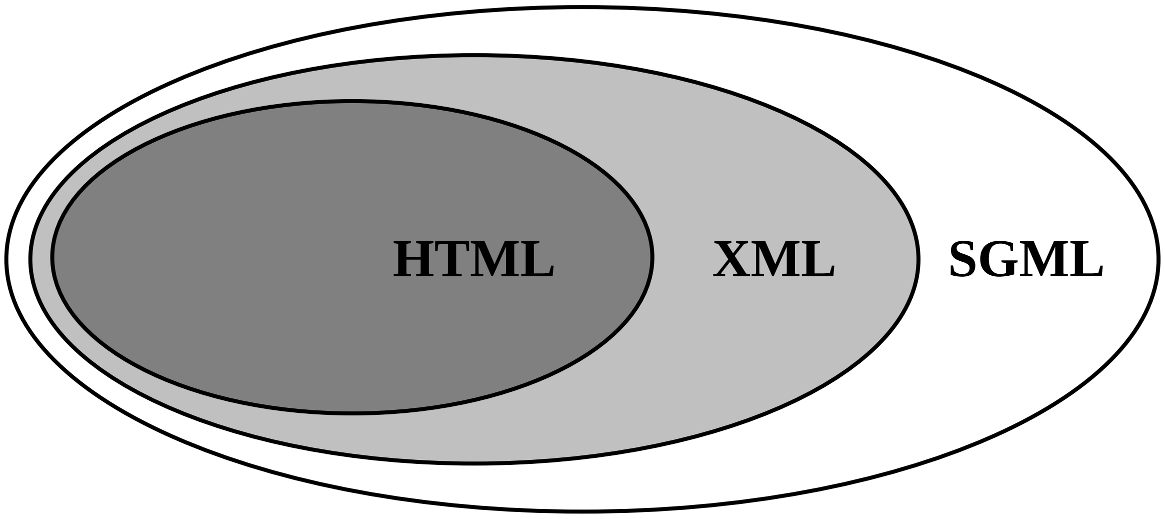


Figure 3.1 The scope of three markup languages

HTML and XML are derived from SGML, but HTML has no flexible. XML gets together the benefits from SGML but keeping flexibility. That's why we adopt XML to be the basis of proxy signature for network data exchange.

3.1 XML Namespaces

We didn't discuss much of XML but only mention some concept related to the thesis. Let's talk directly to the topic of namespaces in XML. The purposes of a namespace when used in an XML document is to prevent the collision of semantically different elements and attributes that happen to have the same name [$BD_1$02]. For example, if someone wants to use the specific tag which is called <CrucialMessage>. This element can refer to the word for commercial use, or it can refer to the secure message for military or national security. Still more problems occurred if this element is created by two different authors. Clearly, the meaning of the element is ambiguous and it is unreasonable to expect any sort of application to be able to make the distinction between the different elements without some other qualifying information.

The problem is solved with the use of a namespace in XML. A namespace in the context of XML is a collection of element or tag names identified by a URI Reference. The usage of an XML namespace is to provide a globally unique name for each tag. A URI Reference is a URI that is used as a string identifier to point to a place providing several tags name. The example we used more often is the namespace used for XML Signature. The URI Reference is like as below:

http://www.w3.org/2000/09/xmldsig#

The collections of element and tag names that are associated with this string identifier include the elements and attributes that help defining an XML Signature. This namespace includes a lot of the tag names for signature algorithms used likes MD5, SHA-1, RSA and so on. More detail of the examples are shown in Chapter 6.

3.2 XML Signature

XML Signature is a rich, complex object. XML Signatures depend on a large number of disparate XML and cryptographic technologies [$BD_2$02]. The XML Signature syntax is for a high degree of extensibility and flexibility; these notions add to the abstract nature of the syntax itself, but provide a signature syntax that is beneficial to almost any signature operation.

The W3C Recommendation, XML-Signature Syntax and Processing, defined the XML Signature syntax and its related processing. This recommendation can be found at the World Wide Web Consortium Web site. The XML Signature Recommendation will likely change in subtle ways as XML Signatures becomes more pervasive. We are, nevertheless, not concerned with the nooks and crannies of the specification, but instead with the basic reason for its existence, examples, and the fundamental properties that define an XML Signature. One might question why we need such a rich signature syntax that differs markedly with our existing signature infrastructure. If we compared an existing messaging syntax, such as PKCS#7, to XML Signatures, we would see so many differences in the intent and implementation of the syntax.

We have to realize some tag used in XML Signature first and list the functions of the tags as below:

Table 3.2.1 XML Signature tags list

Tag Name	Description
<SignedInfo>	The tag <SignedInfo> is the information that is actually signed. It consists of two mandatory processes: validation of the signature over <SignedInfo> block and validation of each Reference digest within <SignedInfo> block.
<Transform>	The optional tag <Transforms> contains one or an list of <Transform> blocks, and all of the <Transform> describe how to figure out the signer data object that being digested.
<CanonicalizationMethod>	The tag <CanonicalizationMethod> is the algorithm used to canonicalize the <SignedInfo> block before digested as part of the signature operation.
<SignatureMethod>	<SignatureMethod> is the algorithm used to convert the canonicalized <SignedInfo> block into the <SignatureValue>. It is a combination of a digest and key dependent algorithm and possibly other algorithms such as padding, for example RSA-SHA1 or DSA-SHA1. The algorithm names are signed to resist attacks based on substituting a weaker algorithm.

< DigestMethod> <DigestValue>	<DigestMethod> object is the algorithm applied to the data after Transforms is applied (if specified) to yield the <DigestValue> block. The signing of the <DigestValue> is an object binding resources content to the signer's key.
<KeyInfo>	<KeyInfo> block indicates the key to be used to verify the signature from signer signed. The existed forms for identification include certificates, key names, and key agreement algorithms. This tag may not available if the signer does not wish to reveal any key information or the information may be known within the application's context and don't need to be represented in XML.

The frequent uses of these tags are listed above and most of all detail information related to XML signature is available in W3C web pages or related publications.

3.3 XML Signature using DSA

W3C organization proposed a standard structure and we can use easily the standard structure for XML signature in DSA. The DSA signature scheme can only be used for signing, and it cannot perform encryption or decryption. So the functions of DSA signature include parameter generation, key generation, and then the signing and verifying operations.

The overall structure uses tags we have mentioned before and we discuss a sort of DSA algorithm. DSA, most of the same way as an RSA signature, uses a public key and private key. The private key is used for generating signature and the public key is used for verifying signature. Because of these computationally expensive tasks, most applications, as matter of fact, use pre-generated DSA parameters or use the DSA pre-assigned key pair under X.509.

The XML Signature Recommendation from W3C has supported for DSA via the URI identifier like this:

<SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#dsa-sha1"/>

We know the XML Signature have two kinds of signature algorithms, and the only difference between these two is the <KeyValue> object. It is designed to hold a raw RSA or DSA public key with child object, tag <RSAKeyValue> and <DSAKeyValue>, respectively. Public keys inside <KeyValue> are represented by their Base64 encoding algorithm. But only we interested this time is <DSAKeyValue> and it has following parameters:

```
<DSAKeyValue>
   <P>…</P>
   <Q>…</Q>
   <G>…</G>
   <Y>…</Y>
</DSAKeyValue>
```

Where <P> and <Q> is for <G> generation; and <Y> is signer’s public key for verification. We don’t need, by the way, the value y if in proxy signature processing. More explicit content about proxy signature in XML will be discussed in Section 6.3.

Chapter 4 Proxy Signature

Proxy signature is a kind of delegating mechanism for use in electronic transaction whatever for commerce or public affairs. It is extended from the conventional signature but possessing of a delegating factor. The document signed by someone stands for lawful effect it is given. It must be handled carefully when signing a document; we should treat equally without discrimination like affixing his/her seal to papers. We can not give our private at will to somebody although he/she is the authorized for carrying out the power. RSA or DSA signature just only provides signed by someone in person, it has no feature for delegation. Only the way to achieve the goal is to give the private key to proxy signer but violating the principle that encountering forge or unauthorized signed if malicious proxy signer to do so. The pure signature is not sufficient to satisfy the requirement currently and it is really a crucial topic to establish a full solution about proxy signature.

To deploy delegation of capability in the electronic world, proxy signature schemes have been proposed. The proxy signature scheme is firstly proposed by Mambo, Usuda and Okamoto [$MUO_1$96, $MUO_2$96] and it is based on discrete logarithms [ElG85] for partial delegation of signing capability. MUO scheme, however, does not provide non-repudiation function of proxy signatures [Zha97, Sun99]. Non-repudiation means signature signers, both the original signer and proxy signers, cannot falsely deny after the signature had been created. In practice, it is important and sometimes necessary to know who the actual signer of the proxy signature is for auditing purpose or when there is abusing of signing capability. Some of theses, therefore, proposed non-repudiated proxy signature scheme [HWW01, LHW98, $LKK_2$01, Sun99, Zha97] which means the signature signers, both original and proxy signers, cannot disavow after the signature is generated. In Section 4.1, it illustrates the characteristics and how many kinds of delegation are available. The next section we discuss what the proxy unprotected or protected scheme is, and the last we will talk about the overview of algorithm we adopted.

4.1 Characteristics of Proxy Signature

We firstly illustrate the properties of digital signature should have and the proxy signature ought to have too. The focal points are not beyond the scope of non-repudiation and unforged property. A digital signature must have many characteristics. We list some of characteristics [Sch00] in the following:

- **The signature is authentic:**

  The signature is convinced by the document's recipient that the signer signed the context deliberately. The system will collapsed and non-meaningful if verifier does not trust this point.

- **This signature is unforgeable:**

  The signature is proved that the signer surely sign the document, and no one else can on behalf of the signer to create the signature. This would be constructed the basic trust relationship between signer and verifier.

- **The signed document is unalterable:**

  After the document is signed, it cannot be modified by any person including the original signer. It will be influenced on all result of signed information if signer changes the original content haphazard.

- **The signature cannot be repudiated:**

  Once the signer signed a signature, he cannot claim that he didn't sign after doing it. This is similar sense with the first point we mentioned.

- **The signature must be verifiable:**

  If a dispute arises due to whether a party signed a document, an trusted third party should be able to resolve the entanglement equitably, without requiring access to the signer's secret information likes private key and so on.

Proxy signatures are signature scheme that an original signer delegates his signing capability to a proxy signer, and then the proxy signer creates a digital proxy signature on behalf of the original signer. The proxy signer maybe not knows what the original signer's private key is and use the proxy information generating the proxy signature from the agreement between original and proxy signer.

According to proxy signature of Mambo et al [$MUO_1$96, $MUO_2$96], there are three kinds of delegation available: full delegation, partial delegation and delegation by warrant. The full delegation, for the security consideration, is barely used.

- **Full delegation:**

  In full delegation, a proxy signer is given the same secret *s* that is the private key of the original signer has. Because of full delegation, the proxy signature created by this proxy signer is indistinguishable from the signature created by the original signer. It is dangerous likes if we hand over the important key to another, but we have no idea what is the matter he/she will do except signing the signature this time.

- **Partial delegation:**

  In partial delegation, a new secret σ is computed from the secret s of an original signer, and σ is given to a proxy signer in a secure way. From security requirement s should not be computed from σ. It is safe if just produced a temporary secret information to proxy signer and that will not be worried the issue of private information leaking. There are two types of signature scheme, moreover, for partial delegation and we talk more in Section 4.2.

- **Delegation by warrant:**

  This kind of delegation is implemented by using a warrant m_w [LK99, Neu93], which certifies that designated proxy signer is exactly the signer to be entrusted. Delegation by warrant is performed by the consecutive execution on signing of the public key signature scheme and that is time-consuming. But, it is appropriate for restricting documents to be signed; a warrant, for example, can state the valid time. In addition, there are two types of scheme for this approach.

(1) Delegate proxy:

The original signer signs a document, declaring someone to be designated as proxy signer under the original signer's secret key by an ordinary signature scheme. The created warrant is given to the proxy signer.

(2) Bearer proxy:

In this type, a warrant is composed of a message part and an original signer's signature for newly generated public key. The secret key for a newly generated public key is given to proxy signer in a secure way.

In the partial delegation, the adopted algorithm is belong to this type, that we can classify proxy signature schemes into designed and non-designed proxy signature schemes according to whether the original signer delegate the power to a specific proxy signer in the proxy key generation phase.

(1) Designated proxy signature:

In this scheme, the original signer specifies the identity of a proxy signer as a form of warrant in proxy generation.

(2) Non-designate proxy signature:

In this scheme, the original signer does not satisfy a proxy signer in the proxy generation phase. Instead she can specify the set of allowed proxy signers of allowed message space.

The partial delegation we have mentioned can be extended into two categories: proxy unprotected and protected scheme. Our proposed algorithm is a kind of proxy protected scheme and that would be safer and more trusted system than which of unprotected one. Section 4.2 is illustrated what the difference and mechanism between those of two schemes.

4.2 Proxy-unprotected and Proxy-protected Scheme

The proxy signature we proposed belongs to a kind of partial delegation, and firstly to understand the basic concepts of two schemes. We put the things in order as following two sub-sections.

4.2.1 Proxy-unprotected Scheme

Besides this proxy signer, the original signer also can create a valid proxy signature. But the third parties who are not designated as a proxy signer cannot create a valid proxy signature of the proxy signer.

The original signer can generate counterfeit information in some times if he/she wants to inflict on the proxy signer. If he/she do so and the verifier accepts the signature, the proxy signer can not prove that is not signed from him/her. It has no guarantee to proxy signer in the session and that will lead to the system causing some argument or a blind spot during transactions.

4.2.2 Proxy-protected Scheme

On the opposite point of view, this scheme is only had the proxy signer to be designated can create a valid proxy signature for the original signer. The third parties and even the original signer cannot create a valid proxy signature of the proxy signer. The scheme totally protects the proxy signer's right and so that no one can forge the proxy information so does the original signer.

Also, we know the partial delegation has a capability to prevent proxy signer derives the original private key from proxy information; it indirectly protect the right of the original signer in order not to let the crucial message leaking out. The proxy protected scheme protects proxy signer's right and they establish, therefore, a mutual protection between original and proxy signer. The verifier can distinguish all situation occurred during sessions and everyone can work properly without any unpredictable condition.

4.3 Overview of proxy protected DSA scheme

The protected scheme of partial delegation adopted this thesis is called "Proxy protected DSA scheme". There is not much of detail algorithms discussed here but the work procedures are briefly talked here. This is a kind of proxy signature extended from DSA signature and it mostly uses the same signing algorithms with using different signing key. This is the DSA signature equations for signature value (r, s) and the notations definition as following:

p, **q**: Two large primes with $q|(p-1)$.

g: A element of order q in Z_p*.

k: A randomly or pseudo-randomly generated integer with $0 < k < q$.

h(m): Message digest of text to be signed.

$r = (g^k \bmod p) \bmod q$

$s = k^{-1}(h(m) + xr) \bmod q$, where x is signer's private key

The proxy protected DSA scheme uses two different notation in those two equations are *g'* and s_B. Here is the signature for the proxy protected DSA scheme:

r = (g'^{k} mod p) mod q

s = $k^{-1}(h(m) + s_B r)$ mod q, where s_B is the proxy key

It has a compatibility with DSA signature because they have the same signing structure with two different parameters. It is made more easier if building up a system compatible with each other, and the next we will briefly discuss the workflow of proxy signature generation and signing.

Here are three steps of proxy protected DSA scheme from generation to verification, and the figures show which parameters will be changed during the specific phase.

This is the phase of proxy generation as below:

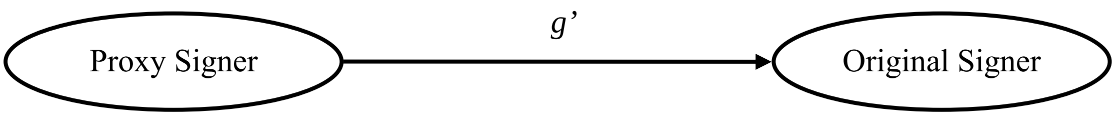


Figure 4.3.1 Proxy generation

The proxy signer creates a parameter *g'* from the original signer's *g* with a random seed. That is a way to avoid proxy signer uses the same parameter the same with DSA signature to create a different value *r*. The generating function is based on difficult in resolving ElGamal discrete logarithm.

After creating *g'*, the original signer should create a temporary signature value for proxy signer. This is the phase of proxy delivery and key generation as below:

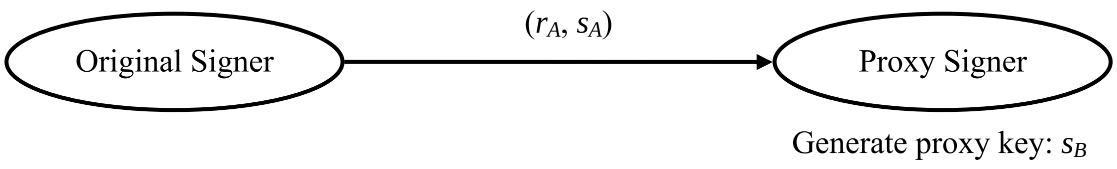


Figure 4.3.2 Proxy delivery and key generation

The original signer generates the temporary signature (r_A, s_A) to proxy signer and it is not for our final signature value sending to verifier. The proxy signer received information from original one and generates the proxy key. We also can say that (r_A, s_A) is a delegating factor and the proxy key is obviously created from this value pair.

The proxy signer already had the proxy key, and he/she starts to make all related parameters for signature. The *g'* we mentioned is a part of signature value and it would be used during verification. There are five parameters needed to be transferred to verifier to verify the document signed by proxy signer under delegated by original signer. Those five parameters are the very proxy signature value. This is the phase of proxy signature verification as below:

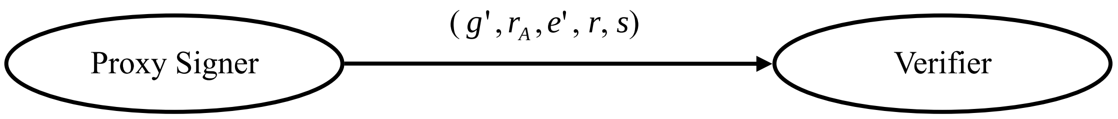


Figure 4.3.3 Proxy signature signing

After verifier gets the signature value and he/she will try to prove the proxy signature is right and truly delegated from original signer. The verifier just prove if v=r and so that he/she validates the signature successfully. We don't illustrate the formulas or notations here moreover and all related procedure. There will have more detail of introduction at next chapter.

Chapter 5 Adopted Algorithm

The proxy signature algorithm we adopted is called “DSA proxy protected scheme” and which is derived by Yeh and Chang [YC05]. It based on and extended from ElGamal and DSA algorithm. DSA proxy protected scheme is a kind of proxy-protected scheme which can justify the role of original signer being honest during the transaction.

The proxy-unprotected scheme, though, is more efficient than the proxy-protected one. The proxy-unprotected scheme, we have mentioned, just can be applicable as the original signer is honest. The proxy-protected schemes also have the ability to prevent cheating motives intrigued by the original signer but no need to verify proxy signers. It is safer than proxy-unprotected one and keeps the privilege for verifiers. The algorithm, also, is more flexible and could be applied to the Public Key Infrastructures.

The notations are defined as follows:

O: An original signer.

P: A proxy signer.

V: A verifier.

p, q: Two large primes with q|(p-1).

g: A element of order q in Z_p*.

h( ): A one-way hash function.

$\mathbf{x_u}$: The secret key of user u.

$\mathbf{y_u}$: The public key of user u.

m: A message to be signed.

A→B: A sends message to B.

There are four sections, proxy generation and delivery, proxy verification and proxy key generation, signed by the proxy signer and verification of the proxy signature in proxy-protected scheme. Let the original signer 'O' has key pair (x_A, y_A) where $y_A = g^{x_A}$ mod p. The value x_A is the original signer's private key and y_A is the original signer public key. It will be illustrated the proxy generation and how we deliver the information between original and proxy signer in Section 5.1.

5.1 Proxy Generation and Delivery

The proxy signer '*P*' selects a random k_0, computes g' and sends it to the original signer. After receiving g', the original signer creates s_A and sends (r_A, s_A). The parameters g' and r_A are public information. The protocol is showed as follows:

P: Select a random, k_0 $(1 \le k_0 \le p-1)$.

Compute $g' = g^{k_0} \bmod p$.

P→ O g'

O: Compute $k_A \in_R Z_q^*$ and $r_A = g^{k_A} \bmod p$.

Compute $e = h(g'^{k_A}) \bmod p$.

Set $s_A = (x_A e + k_A) \bmod q$.

O→ P (r_A, s_A).

Proxy signer obtains the information from original one and extracts r_A, in the first, to be a part of proxy signature.

5.2 Proxy Verification and Proxy Key Generation

The proxy signer checks the validity of (r_A, s_A) and computes a proxy key s_B. The protocol is as follows:

P: Receive (r_A, s_A) from original signer

Check $r_A = g^{s_A} y_A^{-e'}$ mod *p* where $e' = h(r_A^{k_0})$.

If it holds process the follows, else reject it.

Computes $s_B = s_A k_0^{-1}$ mod *q*.

Thus, the proxy key is s_B.

The value r_A has a function to determine whether the proxy information is right or not and it is not merely a part of proxy signature. The proxy signer will create a proxy key s_B from proxy information and the key value is related to s_A. The proxy key, thus, has a factor from the original signer.

Proxy key had been generated and the next step is to demonstrate how to sign the document by using it. There still have some parameters not to be generated, and how we construct a full signature value.

5.3 Proxy Signature Signing

Suppose the original signer will sign on the message *m* as the DSA algorithm using the proxy key s_B. The protocol is the follows:

P: To sign on message m, first compute *h(m)*.

Then select a random $k \in Z_q^*$.

Compute r = (g'^{k} mod p) mod q

Set s $= k^{-1}(h(m) + s_B r)$ mod q

This signing step is similar to the DSA scheme, the only differences are two parameters replacement: *g'* instead of *g* and s_B instead of *x*. The proxy signature we generate is the tuple (g', r_A, e', r, s).

5.4 Verification of the Proxy Signature

To verify the proxy signature (g', r_A, e', *r*, *s*) on message *m*, a verifier should do the following steps:

V: Verify that $1 \le r \le q$ and $1 \le s \le q$; if not, then reject the signature.

Compute $w = s^{-1}$ mod q.

Compute $u_1 = w \cdot h(m)$ mod q, $u_2 = rw$ mod q, and $u_3 = e'u_2$ mod q.

Compute v=($g'^{u_1} r_A^{u_2} y_A^{u_3}$ mod p) mod q.

Accept the signature if and only if v = r.

To verify the proxy signature (g', r_A, e', r, s) on message m, a verifier checks whether v = r, where v=($g'^{u_1} r_A^{u_2} y_A^{u_3}$ mod p) mod q.

In order to prove that the proposed scheme works correctly and explain that the proposed conforms to the requirements of the proxy signature schemes, there are two following theorems being listed:

Theorem 5.1: If the proxy secret (r_A, s_A) is constructed correctly, then it will pass the verification by using $r_A = g^{s_A} y_A^{-e'}$ mod p.

Proof:

Suppose the proxy secret (r_A, s_A) and $g' = g^{k_0}$ mod p is correct. We have

$$s_A = (x_A e + k_A) \bmod q.$$

Then make the substitutions

$$e = h(g'^{k_A}) = h((g^{k_0})^{k_A}) = h((g^{k_A})^{k_0}) = h(r_A^{k_0}) = e' \bmod p.$$

We obtain the following:

$$s_A = (x_A e' + k_A) \bmod q.$$

Rearrange the above equation

$$k_A = (s_A - x_A e') \bmod q.$$

Raise both sides by g

$$g^{k_A} = g^{(s_A - x_A e')} \bmod p,$$

$$r_A = (g^{s_A} \cdot g^{-x_A e'}) \bmod p \ (\because\ r_A = g^{k_A} \bmod p)$$

$$r_A = (g^{s_A} y_A^{-e'}) \bmod p \ (\because\ y_A = g^{x_A} \bmod p)$$

Thus, $r_A = (g^{s_A} y_A^{-e'})$ mod p as required.

We suppose if the proxy signer receives a proxy secret from the original signer correctly before proxy key generation. The proxy signer cannot forge another proxy secret to create a proxy key. It is computationally infeasible generating another r_A to create a valid proxy secret. The original signer, moreover, also cannot forge the proxy key, because the generator has a factor of k_0 generated by the proxy signer. Thus, only the designated signer can create the valid proxy key. The proxy protected DSA scheme, therefore, conforms to the property of *unforgeability*.

Theorem 5.2: If the proxy signature is generated by the proxy signer correctly in the proposed scheme, then it will pass the proxy signature verification.

Proof:

Suppose the proxy signature is correct. It implies that the delegation certification is correct such that we have a valid proxy signature

$$s = k^{-1}(h(m) + s_B r) \bmod q.$$

Rearrange the signature

$$k = s^{-1}(h(m) + s_B r) \bmod q$$

Substitute s_B

$$k = s^{-1}(h(m) + s_A k_0^{-1} r) \bmod q. \ (\because s_B = s_A k_0^{-1} \bmod q)$$

Substitute s_A

$$k = s^{-1}[h(m) + (x_A e + k_A) k_0^{-1} r] \bmod q. \ (\because s_A = (x_A e + k_A) \bmod q)$$

Raise both sides by g'

$$g'^{k} = (g'^{s^{-1}h(m)} \, g'^{k_A k_0^{-1} r s^{-1}} \, g'^{x_A e k_0^{-1} r s^{-1}} \bmod p) \bmod q.$$

Substitute g'^{k} by r, $g'^{k_A k_0^{-1}}$ by r_A and $g'^{x_A k_0^{-1}}$ by y_A

$$r = (g'^{s^{-1}h(m)} \, r_A^{rs^{-1}} \, y_A^{ers^{-1}} \bmod p) \bmod q.$$

Let $w = s^{-1} \bmod q$, $u_1 = w \cdot h(m) \bmod q$,

$u_2 = rw \bmod q$,

$u_3 = e' u_2 \bmod q$.

We yield

$r = (g'^{u_1} r_A^{u_2} y_A^{u_3} \bmod p) \bmod q$ as required.

The proxy signer uses the proxy key to sign on the document, but the verifier need to use the original signer's public key to verify the signature. It put to the proof indirectly that the document is delegated from the original signer. The proxy key is created interactively by original and proxy signer such that from the signature, a verifier can be aware of the original signer who agrees the proxy signer to sign the message. This property is verifiability. From the theorems, the proxy protected DSA scheme we proposed conforms to all of the proxy signature requirements from Section 5.1 to 5.4. For adapting to the DSA, We do not need to add any warrant information to the proposed scheme.

In this chapter, we can see how the proxy protected DSA scheme works and prove that it satisfies all properties that the proxy signature must have. We can be aware that proxy protected DSA scheme has most of characteristics with DSA standard algorithm; the DSA signature, at the same time, is supported by XML we have introduced in Chapter 3. The proxy protected DSA scheme can be modified with some features merely and so that it would be entirely compatible with the current XML signature structure.

There are five tuples used to be the signature value but DSA only has two. The first stuff we should care is how to deal with parameters allocation. We know XML has a flexible mechanism if adding more parameters. It is boggle to us paying more costs to reconstruct the XML proxy signature modified by original one if using DTD or XML schema to extend the feature. Our principle is not to have any DTD or XML schema file defining new tuples but try to dispose them to some location and so that they can be regarded as a normal text or pure string.

Nevertheless, the client program should handle the redundant parameters for proxy signature and extract them out from the text. The appearance of proxy signature looks like the conventional one. We will discuss further at next chapter and develop, on the other hand, simulating program to generate the parameters used in proxy protected DSA scheme.

Chapter 6 XML Proxy Signature Scheme

In this chapter we will discuss the prototype of XML signature and how we modified the original framework to which we could send the XML data with proxy signature information. It can be, so that, easy to transform without changing the XML signature infrastructure because it would be paid more cost on constructing the DTD or XML schema if added some new parameters. It is not, in the other word, worthy to maintain more formats or files than which we already had.

On the other hand, we designed a simulator for parameter generator of XML proxy signature. The simulation is focus on how to generate the parameter we need but not a whole procedure such as proxy generation and verification. We know all parameters and procedure are followed by DSA scheme and only we care is on parameter disposal.

6.1 XML Signature and XML Proxy Signature

XML signature format had been announced by World Wide Web Consortium (W3C) and the document specified XML digital signature processing rules and syntax. XML signatures provide integrity, message authentication, and/or signer authentication services for data of any type. The example shows the structure of signature applied to XML document:

```
<Signature Id="XMLSignatureSample" xmlns="http://www.w3.org/2000/09/xmldsig#">
  <SignedInfo>
    <CanonicalizationMethod
    Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315" />
```

```
        <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#dsa-sha1"/>
        <Reference URI="http://www.w3.org/TR/2000/REC-xhtml1-20000126/">
        <Transforms>
            <Transform
            Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
        </Transforms>
        <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
        <DigestValue>j6lwx3rvEPO0vKtMup4NbeVu8nk=</DigestValue>
    </Reference>
  </SignedInfo>

  <SignatureValue>MC0CFFrVLtRlk=...</SignatureValue>

  <KeyInfo>
    <KeyValue>
        <DSAKeyValue>
            <P>...</P>
            <Q>...</Q>
            <G>...</G>
            <Y>...</Y>
        </DSAKeyValue>
    </KeyValue>
  </KeyInfo>
</Signature>
```

First thing we should consider is how to apply all of the variables of proxy signature into the XML signature context. We know the overall structure of XML signature format should not be changed unless we define a DTD (Document Type Definition) or XML Schema file for the tags we need. Our proposed method is compatible with original defined format and just for its extension. The Digital Signature Algorithm has 2 parameters being called “*r*”, “*s*”. There are formulas for these parameters as below:

$$r = \left(g^{k} \bmod p\right) \bmod q$$

$$s = \left(k^{-1}\left(H(m) + xr\right)\right) \bmod q$$

Where *k* is a random number less than *q* being generated by signer and the verifier is not need to know how many it is. The signature value is consisted of *r* and *s*. We should make integer to octet-stream conversion according to the “I2OSP” operation defined in the RFC 2437 [PKCS1] specification and Base64 encoding algorithm for value expression in XML signature and it will have the same procedure as dealing with proxy signature value generation.

Here is an example for signature value expression:

```
<SignatureValue>
   Mx4psIy9/UY+u8QBJRDrwQWKRaCGz0WOVftyDzAe6WHAFSjMNr7qb2ojq9kdipT
   8Oub5q2OQ7mzdSLiiejkrO1VeqM/90yEIGI4En6KEB6ArEzw+iq4N1wm6EptcyxXxM9
   StAOOa9ilWYqR9Tfx3SW1urUIuKYgUitxsONiUHBVaW6HeX51bsXoTF++4ZI+DjiP
   BjN4HHmr0cbJ6BXk91S27ffZIfp1Qj5nL9onFLUGbR6EFgu2luiRzQbPuM2tPXxyI7GZ
   8AfHnRJK28ARvBC9oi+O1ej20S79CIV7gdBxbLbFprozBHAwOEC57YgJcx+YEjSjcO
   7SBIR1FiUA7pw==
</SignatureValue>
```

Recall the proxy signature scheme we proposed, there are some parameters needed to be placed to suitable position in XML document besides on standard parameters had been defined. Here are the new parameters we need in XML proxy signature as below:

g' : random number generated for proxy generation and delivery

r_A : the specific parameter for proxy generation

e' : hash value being generated by r_A for proxy verification

r : proxy signature value r

s : proxy signature value s

We, on the other hand, just only have the standard parameters of p and q in DSA scheme instead of g and y. The first step we define the parameters under the tag <DSAKeyValue> are only p and q. Tag <P> and tag <Q> can be used both in DSA signature and proxy-protected DSA scheme.

There is another way to dispose the parameter g'. There is no doubt giving the g' value in tag <G> in order to reduce the extra process of proxy signature parameters, but it is ambiguous regarded it as pure signature or proxy signature. It is tradeoff if we do such of this way, there will be having more discussion in section 6.3. The tag <Y> is no longer to be used in XML proxy signature because the public key y is useless for any operation of proxy signature. s_B is the proxy key for signature. We could see the value of s_B can be put in tag <Y> for signature verification, but it has the same confusedness with tag <G>. To avoid it when verification, there should have more mechanism or specific identifier to distinguish whether is proxy signature or not. There will be the same and more discussion in latter section.

So now, the crucial problem is how to give the other parameters to the XML, but first we should generate all parameters for what is needed to create the XML proxy signature. The program is for simulation of parameters generation including original text, digest value, all parameter and converted to signature value.

6.2 XML Proxy Signature Simulation

This program is designed by JAVA programming language, and it has a lot of benefit such as cross OS platform, object oriented programming and so on. It has a more crucial factor is that it can support cryptographic operations like DES, RSA encryption and decryption by using it’s available library code and so do signature function. The algorithm we proposed is not following the DSA signature for all parameters production. Many parameters, therefore, are needed to be generated without using the available library functions. The common stream length used for cryptography is always over 128 bits. We could, however, use the “BigInteger” class to simulate the long stream in JAVA language. All operations related to create the proxy signature are available for this type of class.

The program provides a graphic user interface to show the result we simulated. Here is the program interface as below:

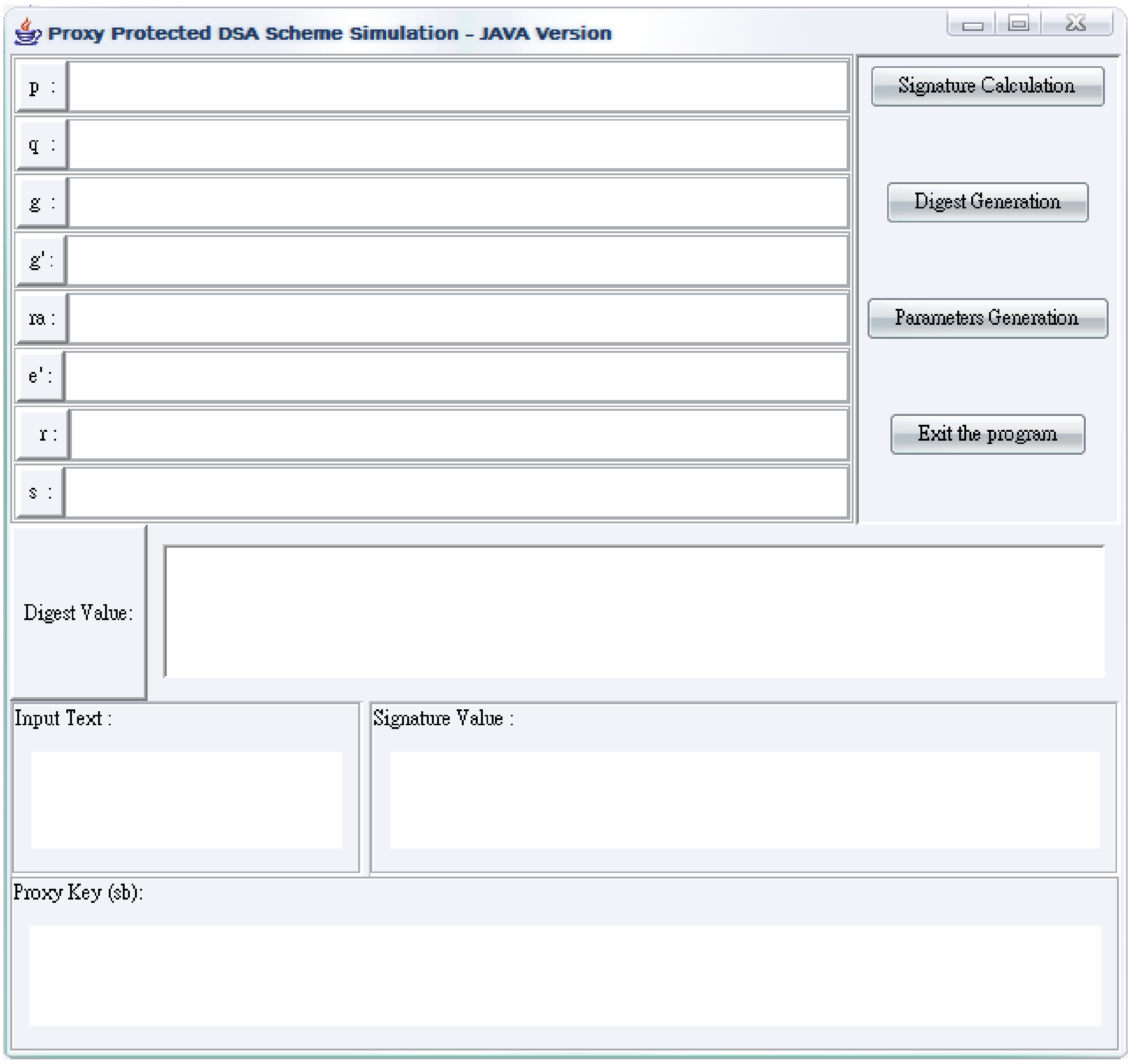


Figure 6.2.1 Program graphic interface

In order to create a proxy signature value, we should do the following steps as below:

Step 1. Input the text needed to be signed.

Step 2. Generate a digest value for inputted text, and the adopted algorithm is SHA-1 which is the same with standard DSA signature.

Step 3. Calculate the tuple (g', r_A, e', r, s) and there is some parameters generated by using the digest value which step 2 is given.

Step 4. Convert the tuple (g', r_A, e', r, s) from binary stream to printable characters using I2OSP and Base64 encoding algorithm. Show the converted string in signature value field.

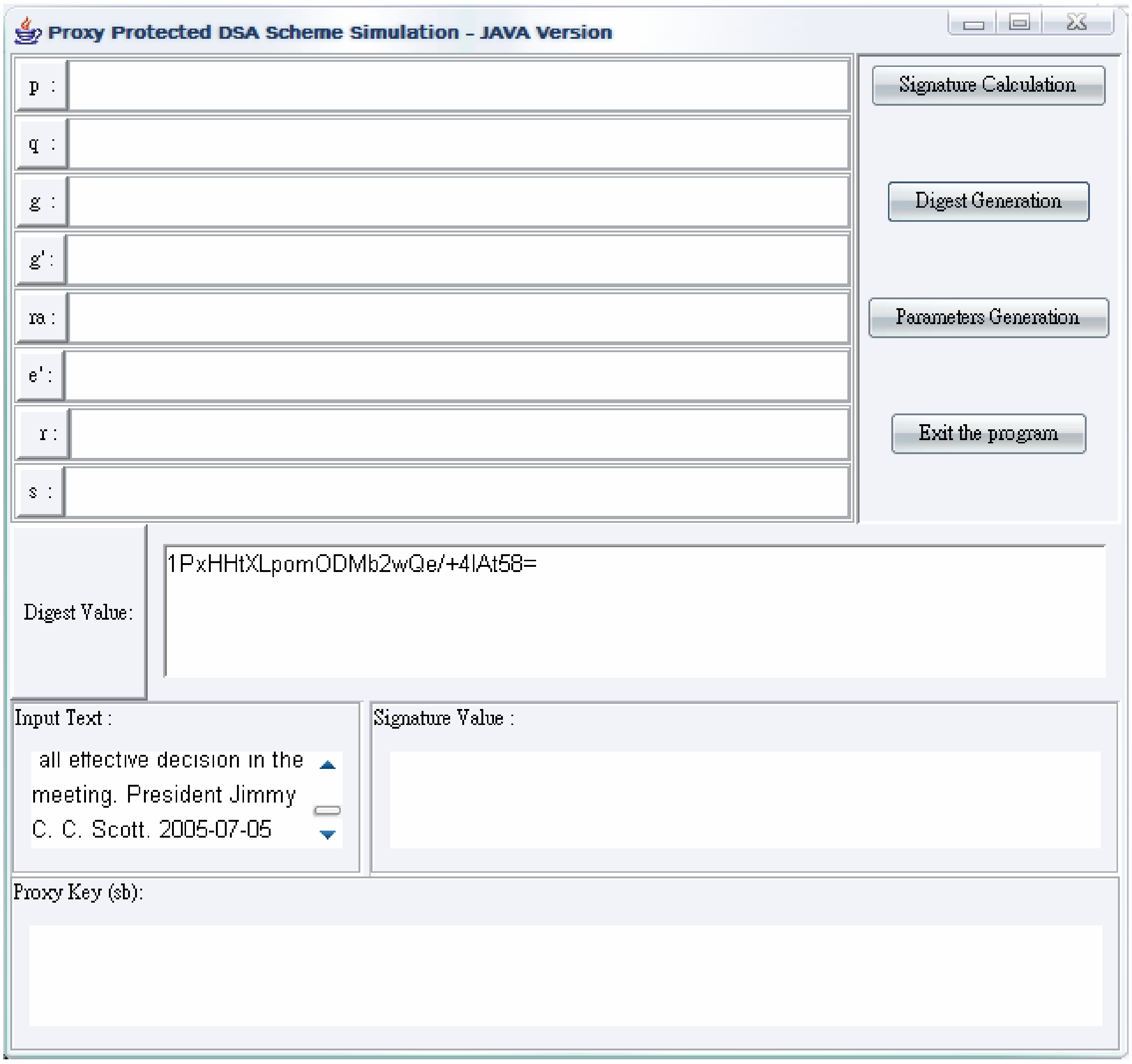
Now, let's implement how to produce the values we need. According to the processing steps mentioned before, the text which is simulated a president authorizes the general manger to take charge of the important meeting and behalf of him/her to make all decision is like following italic words:

"During business trip, I have authorized general manager, Mr. Bob J. Smith, taking charge of the 3rd quarter personnel administration meeting and behalf of me making all effective decision in the meeting. President Jimmy C. C. Scott. 2005-07-05"

The message will be the inputted text and it need to be manipulated by SHA-1. SHA-1 is proposed by Federal Information Processing Standards (FIPS) 180-1 issued by the National Institute of Standards and Technology (NIST) and it is called secure because it is computationally infeasible to find a message which corresponds to a given message digest, or to find two different messages which produce the same message digest. There is a simple class called "MessageDigest" can generate the digest of text easily. We should convert the digest value from stream data to printable characters using base64 encoding. The digest value can be converted and packed the tag <DigestValue> of XML signature as below:

```
<DigestValue>
   1PxHHtXLpomODMb2wQe/+4IAt58=
</DigestValue>
```

The figure is represented the executing result after digest calculation:



1PxHHtXLpomODMb2wQe/+4IAt58=

all effective decision in the meeting. President Jimmy C. C. Scott. 2005-07-05

Figure 6.2.2 Execution of digest calculation

We can generate all parameters related to digest value after creating. On the other hand, the original signer should have a key pair for signature. This simulation will generated a DSA key pair randomly for the original signer use and exchange the random number g' between original and proxy signer. Recall the equation in proxy generation:

$$g' = g^{k_0} \bmod p$$

We know g and p are from DSA key pair generation and k_0 is a random number for session use. So that the g' can be, therefore, obtained by those of two tuples easily. There are still 4 more tuples needed to be generated and the next is r_A. The original signer will produce r_A and s_A to proxy signer during proxy generation and delivery, but s_A is needed for proxy key generation but not for proxy verification. We just only care how to get the r_A and put into the signature value. It can be produced by following equation:

$$r_A = g^{k_A} \bmod p \quad \text{where} \quad k_A \in_R Z_q^*$$

Also, there is no doubt that we get $e' = h(r_A^{k_0})$ by obtaining r_A and the rest of two are r and s. We can see the r and s production are all following DSA signature except for s_B in place of x and g' taking place of g. These are 2 equations for r and s:

$$\mathrm{r} = (g'^{k} \bmod \mathrm{p}) \bmod \mathrm{q}$$

$$\mathrm{s} = k^{-1}(h(m) + s_B r) \bmod \mathrm{q}$$

We know s must have $h(m)$ for generating value and $h(m)$ is the very digest value that just mentioned before. All specific parameters we have generated put together and transfer by I2OSP and base64 encoding. That is, then, a proxy signature value we created for the text.

Citing an instance by the message we just produced at previous pages. Suppose the original signer key is generated by following parameters p, q and g with tags:

<P>
AP1/U4EddRIpUt9KnC7s5Of2EbdSPO9EAMMeP4C2USZpRV1AIlH7WT2NWPq/xfW
6MPbLm1Vs14E7gB00b/JmYLdrmVClpJ+f6AR7ECL
</P>

<Q>
AJdgUI8VIwvMspK5gqLrhAvwWBz1
</Q>
<G>
APfhoIXWmz3ey7yrXDa4V7l5lK+7+jrqgvlXTAs9B4JnUVlXjrrUWU/mcQcQgYC0SR
ZxI+hMKBYTt88JMozIpuE8FnqLVHyNKOCjrh4rs6Z1kW6jfwv6ITVi8ftiegEkO8yk8b
6oUZCJqIPf4VrlnwaSi2ZegHtVJWQBTDv+z0kq
</G>

Original signer private key is simulated and created by the algorithm following the DSA key generation. The producing method is shown as below:

x_A = a randomly or pseudo-randomly generated integer with $0 < x_A < q$

$y_A = g^{x_A} \bmod p$

After obtaining x_A and y_A, we also know that is easy getting r_A by selecting a random number k_A. Here is the value r_A being shown:

<RA>
Lz5gJLvNbAIBMG9t6mrLGOeTQPQclqGbbUHslLZnFeGvygvqHw96cGTSUtFgJ0YK
YOWWyR/tlXlIQjn5YjsWcOL1/CmMUt6Ca2w7siBKJWHGDVcBRRJ9ZElxz1Yowkn
HI5noo5szb4qdC4uE6jQA9ZoUvQEytrlss90F1J3rexc=
</RA>

The g' is also created by random number k_0. This is the g' we create as below:

<GP>

AKCJs8YBU6mZoW9QkevAjEEwH43xACe7eF+Tvj7EWdJDGTqGyhDiyoPx/vyekj0C
IJe690R+CUu5o6uoJbq1YtSqQAWKcf4Oqzs3IrtD3RQ2UwtqabDhwOWHbx8O0wWQ
Y53+T/YBkHgf7ek1ys4qVYak7nZTZmR0M0xQrc3cWcJH

</GP>

The value *e'* is a message digest which is derived from the equation: $e' = h(r_A^{k_0})$. We can use the same way of we produce message digest value to obtain *e'*. The digest value is a fixed length being same with message digest and so that is the value *e'* as below:

<EP>

PtkhMHMhA0WPrReQx0oFfzvRhV0=

</EP>

The value is generated at the same manner with DSA, but we should substitute *g* by *g'*. This is the *r* as below:

<R>

b1qFyy9BGA6tY9up41swQxkYkd0=

</R>

There is more complicate if we would like to generate the *s*, because we should get the signature value of s_B before obtaining *s*. The value s_B also stands for a proxy key for proxy signature use. Let's recall the equation how to generate the s_B, and here is the equation being shown:

$$s_B = s_A k_0^{-1} \bmod q.$$

According to this, we know there has some relationship with s_A and obtain using this way:

$$s_A = (x_A e + k_A) \bmod q \quad \text{where } x_A \text{ is original private key}$$

The value e is produced by original signer and it's also a message digest created by SHA-1 algorithm. Here is the equation:

$$e = h(g^{\prime k_A}) \bmod p$$

The s_B comes out by those of several steps and finally we can generate the value like this:

```
<SB>
   HApBdtx3Yd75tH9en1kyLBVwXDc=
</SB>
```

There is also no different with DSA scheme if we create the s, but we need to use proxy key s_B instead of original signer private key x_A. The message digest $h(m)$ we already got before and we can produce the value like this:

```
<S>
   GAQyBC9c+c+iME5NDvUo5WuxKKk=
</S>
```

After all parameters are acquired smoothly, the final step is that we should generate the proxy signature value. The proxy signature value is concatenated all parameters of g', r_A, e', r and s. We should convert stream by I2OSP function before Base64 encoding algorithm. I2SOP function is recommended by XML Signature Workgroup and operation is defined in the RFC 2437 in PKCS specification. The following manipulating methods are described for how to use the I2OSP function.

Input:

x	nonnegative integer to be converted
l	intended length of the resulting octet string

Output:

X	corresponding octet string of length l; or return message "integer too large"

Steps:

1. If $x \geq 256^l$, output "integer too large" and stop.

2. Write the integer x in its unique l-digit representation base 256:

$$x = x_{l-1}256^{l-1} + x_{l-2}256^{l-2} + \ldots + x_1\, 256^1 + x_0$$

where $0 \geq x\_i < 256$ (note that one or more leading digits will be zero if $x < 256_{l-1}$).

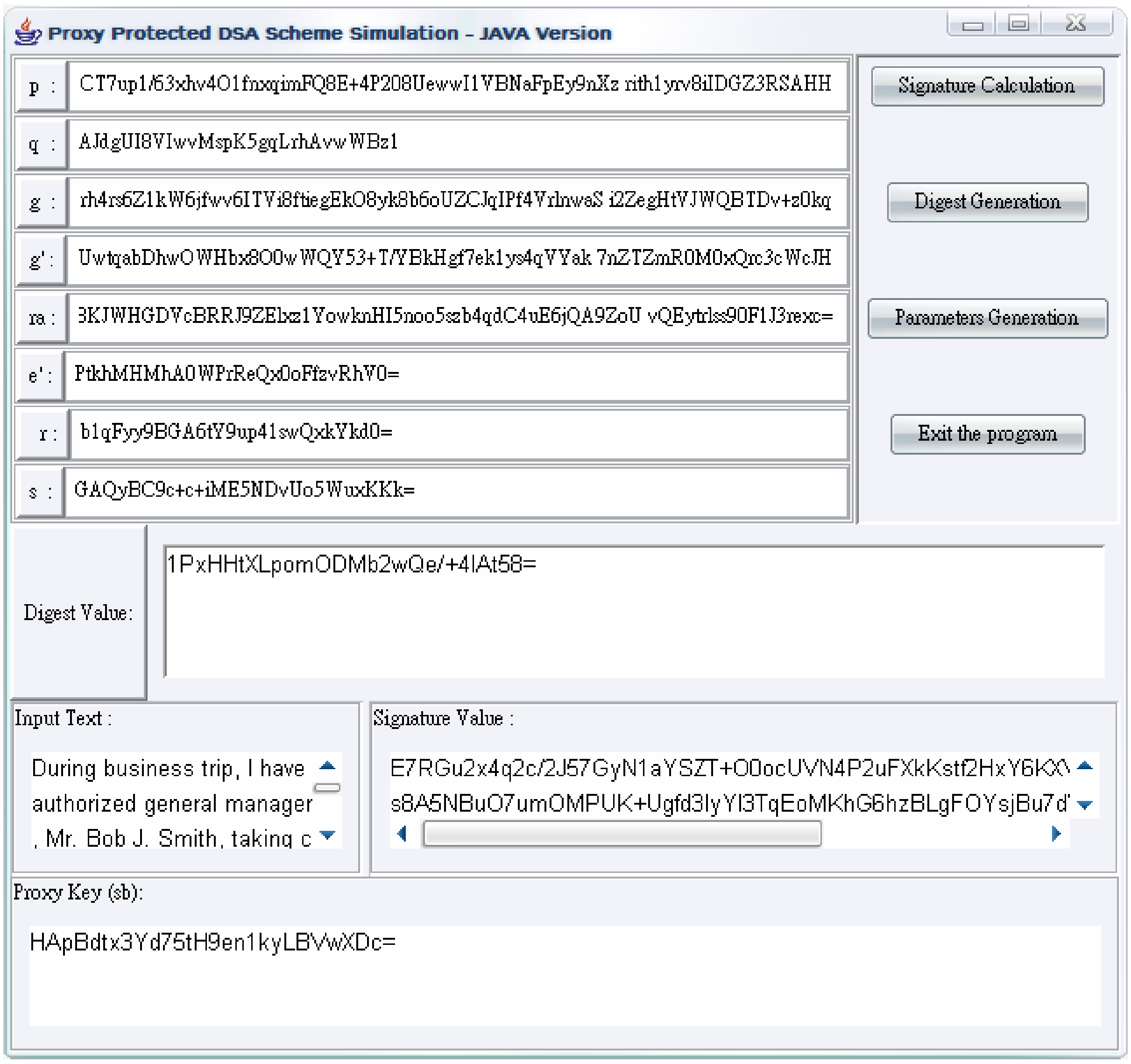
3. Let the octet X_i have the value x_{l-i} for $1 \leq i \leq l$. Output the octet string:

$$X = X_1\, X_2 \ldots X_l.$$

After that, we convert the value by Base64 encoding for representing printable characters format such like the result as below:

<SignatureValue>
E7RGu2x4q2c/2J57GyN1aYSZT+O0ocUVN4P2uFXkKstf2HxY6KXVyZYIFPU20xNnJ
sicto0g3/MLs8A5NBuO7umOMPUK+Ugfd3lyYl3TqEoMKhG6hzBLgFOYsjBu7dVNR
AIbhbJ1p9lTJoLcwWp8RdX0nPk8UXS+nlybUzYieTwTtEa7bHirZz/YnnsbI3VphJlP47S
hxRU3g/a4VeQqy1/YfFjopdXJlggU9TbTE2cmyJy2jSDf8wuzwDk0G47u6Y4w9Qr5SB9
3eXJiXdOoSgwqEbqHMEuAU5iyMG7t1U1EAhuFsnWn2VMmgtzBanxF1fSc+TxRdL6
eXJtTNiJ5E7RGu2x4q2c/2J57GyN1aYSZT+MTtEa7bHirZz/YnnsbI3VphJlP47QTtEa7b
HirZz/YnnsbI3VphJlP47Q=AKCJs8YBU6mZoW9QkevAjEEwH43xACe7eF+Tvj7EWd
JDGTqGyhDiyoPx/vyekj0CIJe690R+CUu5o6uoJbq1YtSqQAWKcf4Oqzs3IrtD3RQ2Uw
tqabDhwOWHbx8O0wWQY53+T/YBkHgf7ek1ys4qVYak7nZTZmR0M0xQrc3cWcJH
AKCJs8YBU6mZoW9QkevAjEEwH43xACe7eF+Tvj7EWdJDGTqGyhDiyoPx/vyekj0C
IJe690R+CUu5o6uoJbq1YtSqQAWKcf4Oqzs3IrtD3RQ2UwtqabDhwOWHbx8O0wWQ
Y53+T/YBkHgf7ek1ys4qVYak7nZTZmR0M0xQrc3cWcIAoImzxgFTqZmhb1CR68CM
QTAfjQCgibPGAVOpmaFvUJHrwIxBMB+NAKCJs8YBU6mZoW9QkevAjEEwH40=
</SignatureValue>

All above we have mentioned is how to generate the value we need from the simulating program. As the program produce the message digest by inputted text and click the “Digest generation” button, the digest value would be came out on screen. We also click the “Parameter generation” button and all parameters can be shown by each field. This is the result of application interface showing up:



p : CT7up1/63xhv4O1fnxqimFQ8E+4P208UewwI1VBNaFpEy9nXz rith1yrv8iIDGZ3RSAHH

q : AJdgUI8VIwvMspK5gqLrhAvwWBz1

g : rh4rs6Z1kW6jfwv6ITVi8ftiegEkO8yk8b6oUZCJqIPf4VrlnwaS i2ZegHtVJWQBTDv+z0kq

g' : UwtqabDhwOWHbx8O0wWQY53+T/YBkHgf7ek1ys4qVYak 7nZTZmR0M0xQrc3cWcJH

ra : 3KJWHGDVcBRRJ9ZElxz1YowknHI5noo5szb4qdC4uE6jQA9ZoU vQEytrlss90F1J3rexc=

e' : PtkhMHMhA0WPrReQx0oFfzvRhV0=

r : b1qFyy9BGA6tY9up41swQxkYkd0=

s : GAQyBC9c+c+iME5NDvUo5WuxKKk=

Digest Value: 1PxHHtXLpomODMb2wQe/+4IAt58=

Input Text :
During business trip, I have
authorized general manager
, Mr. Bob J. Smith, taking c

Signature Value :
E7RGu2x4q2c/2J57GyN1aYSZT+O0ocUVN4P2uFXkKstf2HxY6KX\
s8A5NBuO7umOMPUK+Ugfd3lyYl3TqEoMKhG6hzBLgFOYsjBu7d'

Proxy Key (sb):
HApBdtx3Yd75tH9en1kyLBVwXDc=

Figure 6.2.3　The result of executing sample

Because the proxy signature has more extra values than which used in standard DSA algorithm, it is naturally longer string length than DSA signature value has. Consider more flexible and compatible characteristics as constructing the parameters disposal. There will have many ways to put the parameters to appropriate position. Firstly, we directly perceived through the senses to arrange the signature value in such manner. The other ways to deal with is to take apart some parameters into data block and that is better if more compatible with DSA. We will have more detail discussions at next section.

6.3 Parameters Dispatch

We have mentioned before that we keep all parameters to appropriate position. This is the example using our simulation result:

```
<Signature Id="BusinessTrip" xmlns="http://www.w3.org/2000/09/xmldsig#">
  <SignedInfo>
    <CanonicalizationMethod
    Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315" />
    <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#dsa-sha1"/>
    <Reference Type="http://www.w3.org/TR/2000/09/xmlsig#Object"
    URI="#TripAnnounce">
      <Transforms>
        <Transform
        Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
      </Transforms>
      <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
      <DigestValue>1PxHHtXLpomODMb2wQe/+4IAt58=</DigestValue>
    </Reference>
  </SignedInfo>

  <SignatureValue>E7RGu2x4q2c/2J ... kevAjEEwH40=</SignatureValue>

  < Object>
    <CrucialMessage id ="TripAnnounce" >
      During business trip, I have authorized general manager, Mr. Bob J. Smith, taking
```

```
        charge of the 3rd quarter personnel administration meeting and behalf of me making
        all effective decision in the meeting. President Jimmy C. C. Scott. 2005-07-05
      </CrucialMessage>
    </Object>

    <KeyInfo>
      <KeyValue>
        <DSAKeyValue>
          <P> AP1/U4EddR…6AR7ECL </P>
          <Q> AJdgUI8VIwvMspK5gqLrhAvwWBz1</Q>
          <G> APfhoIXWmz3ey7yrXDa…HtVJWQBTDv+z0kq </G>
        </DSAKeyValue>
      </KeyValue>
    </KeyInfo>
  </Signature>
```

This is the direct sense to establish the proxy signature, and there is not having any extra tag we should define. If we construct the XML proxy signature with XML signature under PKI (X.509), the signature value would have a different length of string. In the other word, the verifier has no idea what kind of signature he/she received except identify the string length of signature. It is not a good idea to integrate both of two signature algorithm into X.509 system. The goal we want to achieve is to have the most of compatibility as possible, just only put the value *r*, *s* in signature value and the others will be arranged to suitable place.

There is no any other tag can be used for proxy signature besides on “Object” block. We know the “Object” block is for the text to be signed and all data here would be regarded as pure string. In order to fill the bill with XML signature using DSA scheme, we put all of extra parameters to object block. There is not any different with original context externally and the mechanisms of X.509, therefore, would not care what kind of signature it is. We could have some alternative process in client program dealing with it if needed. Take the previous sample for example, g', r_A and e' are extra parameters for proxy signature. We insert these three values into object block like this:

```
< Object>
  <GP>
    AKCJs8YBU6mZoW9QkevAjEEwH43xACe7eF+Tvj7EWdJDGTqGyhDiyoPx/vyekj
    0CIJe690R+CUu5o6uoJbq1YtSqQAWKcf4Oqzs3IrtD3RQ2UwtqabDhwOWHbx8O0
    wWQY53+T/YBkHgf7ek1ys4qVYak7nZTZmR0M0xQrc3cWcJH
  </GP>
  <RA>
    Lz5gJLvNbAIBMG9t6mrLGOeTQPQclqGbbUHslLZnFeGvygvqHw96cGTSUtFgJ0Y
    KYOWWyR/tlXlIQjn5YjsWcOL1/CmMUt6Ca2w7siBKJWHGDVcBRRJ9ZElxz1Yo
    wknHI5noo5szb4qdC4uE6jQA9ZoU vQEytrlss90F1J3rexc=
  </RA>
  <EP>
    PtkhMHMhA0WPrReQx0oFfzvRhV0=
  </EP>
  <CrucialMessage id ="TripAnnounce" >
    During business trip, I have authorized general manager, Mr. Bob J. Smith, taking
    charge of the 3rd quarter personnel administration meeting and behalf of me making all
    effective decision in the meeting. President Jimmy C. C. Scott. 2005-07-05
  </CrucialMessage>
</Object>
```

At this moment, we consider how about to replace the value *g* by *g'* and so that reducing redundant tag production. There is, as matter of fact, no problem if we substitute *g* by *g'*. The value *g'* is more meaningful than *g* for proxy signature verification and *g* is no used any more in this session. But thinking over the compatible characteristic again, the value *g* is created from x_A and *g'* is from k_0. How do we recognized the tag <G> is surrounded with *g* or *g'*? Although the format is compatible with normal XML signature, we have to face ambiguous when reading the value of tag <G>. It is not cost much if we have additional tag <GP> standing for value *g'* in the object block, and tag <GP> is merely a pure string for object block. The client program should read these three parameters in order to verify the message.

Moreover, we can put these three parameters into the tag <CrucialMessage> on this example. There should be more secure than previous one, because digest value is consist of SHA-1 conversion of text to be signed and those tags with theirs value. It would be harder and more complicated if tried to forge the signature and value. Here is the tag <CrucialMessage> likes as below:

```
<CrucialMessage id ="TripAnnounce" >
  <GP>
    AKCJs8YBU6mZoW9QkevAjEEwH43xACe7eF+Tvj7EWdJDGTqGyhDiyoPx/vyekj
    0CIJe690R+CUu5o6uoJbq1YtSqQAWKcf4Oqzs3IrtD3RQ2UwtqabDhwOWHbx8O0
    wWQY53+T/YBkHgf7ek1ys4qVYak7nZTZmR0M0xQrc3cWcJH
  </GP>
  <RA>
    Lz5gJLvNbAIBMG9t6mrLGOeTQPQclqGbbUHslLZnFeGvygvqHw96cGTSUtFgJ0Y
    KYOWWyR/tlXlIQjn5YjsWcOL1/CmMUt6Ca2w7siBKJWHGDVcBRRJ9ZElxz1Yo
    wknHI5noo5szb4qdC4uE6jQA9ZoU vQEytrlss90F1J3rexc=
  </RA>
  <EP>
    PtkhMHMhA0WPrReQx0oFfzvRhV0=
  </EP>
  <Text>
    During business trip, I have authorized general manager, Mr. Bob J. Smith, taking
    charge of the 3rd quarter personnel administration meeting and behalf of me making all
    effective decision in the meeting. President Jimmy C. C. Scott. 2005-07-05
  </Text>
</CrucialMessage>
```

Another way we can do is to move parameter “id” in tag <CrucialMessage> to tag <Object>. Thus, three parameters will be included within message and produce new digest value. In the other word, The signature value will be changed too. But one stuff would be cared is that client program should have to parse tags of *g’*, r_A and *e’* out of the tag <CrucialMessage>. It is a tradeoff whether if you want more secure or efficiency for this application.

It is not only focused on parameters dispatch but even more topic about how is the data packed in the XML as well. We know there are many type of data being existed in different form such like WORD, PDF, HTML with photographs and so on. It just only can be accepted by printed characters format in XML context. We still need some mechanisms to conform to all requests although the binary stream can be transferred by Base64 encoding algorithm. In next section, we will implement all cases of different type of data and compare the properties each other.

6.4 XML Proxy Signature Packaging

We will classify three kind of data format firstly before going deep into the topic and these are pure text, binary data, and markup language document:

- **Pure Text:**

  Printable character readable and meaningful for certain human language. Ex. ASCII, BIG-5 …

- **Binary Data:**

  Documentations or graphic files are encoded and needed to be opened by certain application. Ex. WORD, jpeg format picture…

- **Markup Language Document:**

  All fields are represented and surrounded by tags. Ex. HTML, XML…

Different kind of data has different characteristic when constructing XML proxy signature and that is why we have to divide into those of three categories. After that, Let's get into and discuss three type of packaging methods: Enveloped, enveloping and detached signature.

6.4.1 Enveloped Signature

The signature is included in the XML or some sort of markup language document that contains the signature as an element. The content provides the root XML document element.

Obviously, enveloped signatures must take care not to include their own value in the calculation of the tag <SignatureValue>.

```
<!-- Enveloped Signature -->
<OriginalDocument>
   <Signature> ... </Signature>
</OriginalDocument>
```

According to the structure above, the signature is a part of XML document. But how do we know where is the portion of text to be signed? It would have some method to figure out and "XPath" is recommended to apply to this kind of signature. There is going without saying that it can be applied also. Recall the example and transfer to these form:

```
<CrucialMessage>
   <GP>
      AKCJs8YBU6mZoW9QkevAjEEwH43xACe7eF+Tvj7EWdJDGTqGyhDiyoPx/vyekj
      0CIJe690R+CUu5o6uoJbq1YtSqQAWKcf4Oqzs3IrtD3RQ2UwtqabDhwOWHbx8O0
      wWQY53+T/YBkHgf7ek1ys4qVYak7nZTZmR0M0xQrc3cWcJH
   </GP>
   <RA>
      Lz5gJLvNbAIBMG9t6mrLGOeTQPQclqGbbUHslLZnFeGvygvqHw96cGTSUtFgJ0Y
      KYOWWyR/tlXlIQjn5YjsWcOL1/CmMUt6Ca2w7siBKJWHGDVcBRRJ9ZElxz1Yo
      wknHI5noo5szb4qdC4uE6jQA9ZoU vQEytrlss90F1J3rexc=
   </RA>
   <EP>
```

```
    PtkhMHMhA0WPrReQx0oFfzvRhV0=
  </EP>
  <Text>
    During business trip, I have authorized general manager, Mr. Bob J. Smith, taking
    charge of the 3rd quarter personnel administration meeting and behalf of me making all
    effective decision in the meeting. President Jimmy C. C. Scott. 2005-07-05
  </Text>

  <Signature Id="BusinessTrip" xmlns="http://www.w3.org/2000/09/xmldsig#">
    …
    <Transform Algorithm="http://www.w3.org/TR/1999/REC-xpath-19991116">
      <XPath>
        not(ancestor-or-self:dsig:Signature)
      </XPath>
    </Transform>
      …
  </Signature>
</CrucialMessage>
```

The only difference between signature and proxy signature is that proxy signature regards the three parameters as the part of signed message, because XPath function figure out to sign the all text but excluding tag <Signature> and it's content. That would have different signature and digest value if these three parameters are moved out of the tag <CrucialMessage> and put back to the <Signature block>. The values g', r_A and e' are independent with digest value, so we can generate those of three parameters first and then create digest value. These three parameters can't place out of tag <Signature> if they have mutual relationship with digest value. If r_A, for example, has an element of digest value, but it can't be created earlier than digest value. Digest value can't be generated because the absence of r_A and that's why those of three parameters must have no relationship with digest value.

This type of proxy signature is suitable for the signed document adopting XML or HTML form. The signed document can be applied to the internet for some web-based application. It also can be display on common web browser. The drawback of enveloped signature is that the other kinds of data format could not be used, because we have no idea how to pack the signature in the message or binary data.

6.4.2 Enveloping Signature

The signature is over content and found within a tag <Object> of the signature itself. The <Object> block (or its content) is identified via a tag <Reference>.

This is the model of enveloping signature as below:

```
<!-- Enveloping Signature -->
<Signature>
  <OriginalDocument>
    …
  </OriginalDocument>
</Signature>
```

The outer structure is the proxy signature body and we just put the message to be signed in <Object> block. The example we discuss is shown in the beginning and it belongs to of this type of structure. The <Object> block is always put into the pure text and binary data but need some conversion. The Base64 encoding algorithm is the way to transfer the binary data to printable characters.

At this point, what we should concern is that if the binary data is so large that the signature is almost filled with transferred content. It is not an appropriate way to make a proxy signature for huge binary data. Pure text is suitable for enveloping signature and so is markup language document. But most of markup language documents are applicable for enveloped signature in order that being shown directly. The enveloping signature, however, is like a kind of integrated file bundled with signature and original document. It is convenient if designing a program for generation and verification because all information put together and extract easily. The program we designed is based on this reason to develop.

6.4.3 Detached Signature

The signature is over content and out of the tag <Signature>. The target is figured out via a URI. The signature is certainly "detached" from the content as it signed. This definition typically applies to separate data objects, but it also includes the instance where the signature block and data object reside within the same XML document but are sibling elements.

This is the model for detached signature as below:

```
<!-- Detached Signature -->
<Signature>
   …
</Signature>
```

The content is excluded from the proxy signature but maybe exists somewhere. The problem we suffered at first is how to deal with the redundant parameters. The reference points to the destination for signing and it is impossible changing its original content. Maybe adding them into context if the target is text, document or some sort related to readable file. There is no doubt that can not insert them if the target is a type of image or binary file. It should be have more complicate means than the others. But now, let's see the example as below:

```
<Signature Id="BusinessTrip" xmlns="http://www.w3.org/2000/09/xmldsig#">
  <SignedInfo>
    <CanonicalizationMethod
    Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315" />
    <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#dsa-sha1"/>
    <Reference URI="http://www.jimmyscott.com/CrucialMessage.tif">
      <Transforms>
        <Transform
        Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
      </Transforms>
      <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
      <DigestValue>1PxHHtXLpomODMb2wQe/+4IAt58=</DigestValue>
    </Reference>
  </SignedInfo>

  <SignatureValue>E7RGu2x4q2c/2J ... kevAjEEwH40=</SignatureValue>
  <KeyInfo>
    <KeyValue>
      <DSAKeyValue>
        <P> AP1/U4EddR…6AR7ECL </P>
        <Q> AJdgUI8VIwvMspK5gqLrhAvwWBz1</Q>
        <G> APfhoIXWmz3ey7yrXDa…HtVJWQBTDv+z0kq </G>
      </DSAKeyValue>
    </KeyValue>
  </KeyInfo>
</Signature>
```

The original signed object, we know, is used by URI to point out somewhere likes enveloping signature but the content is not existed in local files. There must be have a modified manner and the simplest way is to transfer original binary content by Base64 encoding algorithm. Once the content is converted to printable character, the format will be transformed to the style of enveloping signature. Those of three parameters, thus, can be put together with the content. The example is like as below:

```
<SignedInfo>
  <Reference>
    <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
    <DigestValue>OLaPA/jptmnzqePsSAiqPhFXQBg=</DigestValue>
  </Reference>
</SignedInfo>
```

```
<Object>
  <GP>
    AKCJs8YBU6mZoW9QkevAjEEwH43xACe7eF+Tvj7EWdJDGTqGyhDiyoPx/vyekj
    0CIJe690R+CUu5o6uoJbq1YtSqQAWKcf4Oqzs3IrtD3RQ2UwtqabDhwOWHbx8O0
    wWQY53+T/YBkHgf7ek1ys4qVYak7nZTZmR0M0xQrc3cWcJH
  </GP>
  <RA>
    Lz5gJLvNbAIBMG9t6mrLGOeTQPQclqGbbUHslLZnFeGvygvqHw96cGTSUtFgJ0Y
    KYOWWyR/tlXlIQjn5YjsWcOL1/CmMUt6Ca2w7siBKJWHGDVcBRRJ9ZElxz1Yo
    wknHI5noo5szb4qdC4uE6jQA9ZoUvQEytrlss90F1J3rexc=
  </RA>
  <EP>
    PtkhMHMhA0WPrReQx0oFfzvRhV0=
  </EP>

  <CrucialPicture Id="ImportantPicture" MimeType="image/gif"
  Encoding="http://www.w3.org/2000/09/xmldsig#base64">
    aWcgQmxha2UncyBBdXRoZW50aWNhdGlvbiBTZXJ2aWNlMRQwEgYDVQQLEw
    tFbmdpbmVlcmluZzEWMBQGA1UEAxMNQmlnIEJhZCBCbGFrZTEcMBoGCSqG
    SIb3DQEJARYNYmJiQGJiYmFzLmNvbTAeFw0wMDA2MjAyMTEzMzVaFw0xMT
    A2MDMyMTEzMzVaMH4xCzAJBgNVBAYTAlVTMRMwEQYDVQQIEwpTb21lL
    VN0YXRlMQ8wDQYDVQQKEwZTZXJ2ZXIxFDASBgNVBAsTC1NlcnZlciBDZX
    J0MRMwEQYDVQQDEwpTZXJ2ZXJDZXJ0MR4wHAYJKoZIhvcNAQkBFg9zZXJ
    2ZXJAY2VydC5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMg7Y
    9ZByAKLTf4eOaNo8i5Ttge+fT1ipOpMB7kNip+qZR2XeaJCiS7VMetA5ysX7deDU
```

```
        YYkpefxJmhbL2hO+hXj72JCY0LGJEKK4eIf8LTR99LIrctz
    </CrucialPicture>
</Object>
```

In this case, it will not be a detached signature model but enveloping one whatever it is reasonable and works. We, however, discard this method to establish the proxy signature because of incorrect structure. All about this, there would have another way to fit with our request.

We should not, on the other hand, add parameters any more to follow the compatibility we emphasized in advance but parameters block can be regard as an independent document. Thus, the XML standard announces that the tag <Manifest> could pack multiple objects, so that both of two can be signed at the same time.

```
<Signature Id="DetachedSignature">
    <SignedInfo>
        <Reference URI="#TripSig" Type="http://www.w3.org/2000/09/xmldsig#Manifest">
            <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
            <DigestValue>545x3rVEyOWvKfMup9NbeTujUk=</DigestValue>
        </Reference>
    </SignedInfo>

    <Object>
        <Manifest Id="TripSig">
            <Reference Type="http://www.w3.org/TR/2000/09/xmlsig#Object"
            URI=”#TripParameter”>
              <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
              <DigestValue>3K2DgPfmmrm632wx547nDOU60v8=</DigestValue>
            </Reference>
            <Reference URI="http://www.jimmyscott.com/announce.png">
              <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
              <DigestValue>ALIOvKHxUqjCynnxeBibMJ1Yh5E=</DigestValue>
            </Reference>
        </Manifest>
        <TripParameter>
            <GP>
              AKCJs8YBU6mZoW9QkevAjEEwH43xACe7eF+Tvj7EWdJDGTqGyhDiyoPx/v
              yekj0CIJe690R+CUu5o6uoJbq1YtSqQAWKcf4Oqzs3IrtD3RQ2UwtqabDhwOW
              Hbx8O0wWQY53+T/YBkHgf7ek1ys4qVYak7nZTZmR0M0xQrc3cWcJH
            </GP>
            <RA>
              Lz5gJLvNbAIBMG9t6mrLGOeTQPQclqGbbUHslLZnFeGvygvqHw96cGTSUtF
              gJ0YKYOWWyR/tlXlIQjn5YjsWcOL1/CmMUt6Ca2w7siBKJWHGDVcBRRJ9
              ZElxz1YowknHI5noo5szb4qdC4uE6jQA9ZoUvQEytrlss90F1J3rexc=
            </RA>
            <EP>
              PtkhMHMhA0WPrReQx0oFfzvRhV0=
            </EP>
        </TripParameter>
    </Object>
        ...
```

```
</Signature>
```

The above example has two parts in <Mainfest> block, first one is stored the parameters section named <TripParameter> locally and another is pointed by URI out there. The signature has a final digest value for the <Manifest> to prevent someone from falsification. Thinking more about the parameters section surrounded by the tag <TripParameter>, it looks like using the way of enveloping signature as arranging the original content. We can make it more similar with the detached signature, getting rid of the parameters section and stored to an individual file. Taking the first <Reference> block and figured out by URI same with the second one. Supposed the parameters file is located on the following address: “http://www.jimmyscott.com/proxysigner/thirdqrtsig.xml”, the <Reference> block should be written as below:

```
<Reference URI=” http://www.jimmyscott.com/proxysigner/thirdqrtsig.xml”>
  <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
  <DigestValue>3K2DgPfmmrm632wx547nDOU60v8=</DigestValue>
</Reference>
```

After introducing how to create the detached proxy signature, let’s think about what type of data format suitable for it. There are, as matter of fact, all can be applied to the detached proxy signature. It merely uses the URI but exclude the original content to specify where it is, so that all we do is to get apart the content from the <Signature> block and put into the file anywhere that everybody can read it. But one stuff we concerned is more appropriate than other two data format and that is the binary data. It can be adopted by the detached manner, and the content transferred by Base64 encoding for applying enveloped or enveloping proxy signature will make the XML file too large to read conveniently for human beings. That's why we take the detached proxy signature to be the best way if processing the binary content.

Beyond these three types of proxy signature format, we summarized all characteristics discussed previously at the next section.

6.5 Proxy Signature Packaging Comparison

We have just discussed three types of packing manner for proxy signature and every type can be applied successfully. There is no problem if the DSA protect scheme applied to the X.509 infrastructure because all factors of compatibility have been considered.

All characteristics are enumerated under and pick out one of the type suiting for the original content to be signed:

Table 6.5.1 Three type of signature comparison

Type / Characteristic	**Enveloped Proxy Signature**	**Enveloping Proxy Signature**	**Detached Proxy Signature**
Appropriate data format	Markup language documents like HTML or XML	Pure text	All, but especially better for binary data
Original content location	Surrounded with proxy signature	Inside the proxy signature	Some place and pointed out by URI
Content visible in XML	Visible, can be figured to all or part of the document	Original content and visible in XML	Invisible, exist in some place
Parameters location	Outside the signature and to be the part of original message	Put all together with original message in the proxy signature	Located out of the signature and it can be put together with original message location or not

Now we have accomplished the introduction to the DSA protect scheme we adopted, the overall structure of signature and proxy signature, parameters dispatch, XML proxy signature packaging and comparison. We, at the same time, have proved proxy protected DSA scheme is feasible and compatible with DSA signature under X.509. There should have, somehow, more issues we denoted to, likes how to implement the scheme to PKI practically and so on.

Beyond all of these discussions, Getting into the topic of future work needed to be paid more efforts in the chapter 8. We also summarized conclusion at next chapter and illustrate what the functions of the mechanism it can be done.

Chapter 7 Conclusion

XML is the most popular industry standard over the world, there are many software or network related application adopted this type of language. Different kind of data can not be exchange because of different data format. Between the heterogeneous software which share data mutually if they have the same kind of source. XML would be the best and first choice for use; and it, also, is a flexible and readable language for machine and mankind. There are, on the other hand, many formats had been generally acknowledged like: math, web application, DOM and so on. We didn’t reconstruct all but using current standard, and just modify something to correspond to our requirement.

Proxy signature is more applicable and practicability than pure signature and proxy protected DSA scheme is more compatible than any other. The algorithm we adopted can be coexisted and manipulated with current DSA signature under X.509 structure. The structure is almost likely which of conventional signature because redundant parameters are coped with hidden data; therefore, our proposed method makes both of two exteriors identical.

We combine those of two useful but independent issues and also take their advantages to establish the XML proxy signature system. XML signature is available for years but it is still not sufficient to fit requirement currently if performing practically. Proxy signature is extended, but more applicable and useful than conventional one. A practical proxy system is widely adopted in companies or government nowadays; we, of course, need a resemble system in the virtual space taking the place of it. That why we use this kind of algorithm.

The advantage of proxy protected scheme is that the original signer gives a temporary secret to proxy signer but can not deduce back to the original private key streams. The original signer can produce many temporary secrets for many proxy signers delegating the authority. All proxy signers also prove themselves the delegation whether is valid or not.

If we adopt proxy unprotected scheme, the original signer also can create a valid signature. It is a dangerous if the original is a forger making a fake signature. This is not appropriate application to carry out. Everyone in the electronic transaction should be protected well anytime. That's why we use protected scheme adopted in our mechanism.

Here we have a table comparing the function of some kind of algorithms. This is the table as following:

Table 7.1 Comparison of algorithms

Type / Characteristic	DSA Algorithm	XML Signature	Adopted Mechanism
Verifiability	V	V	V
Strong Un-forgeability			V
Strong Identifiability			V
Strong Undeniability			V
Limited Delegation			V
Forward Security			V
Network Access		V	V
X.509 Support		V	V
Protected Scheme			V
Compatibility		V	V

Type / Characteristic	DSA Algorithm	XML Signature	Adopted Mechanism
Non-Extra Definition		V	V
XML support		V	V

We can see our adopted mechanism has all of listed advantages and just based on present system. Although current proxy signature algorithms have a concrete system or platform supported. If we supposed they can be used in XML system, they still have to re-define the structure and cannot compatible with X.509. Our propose mechanism works fine and fully compatible with present system without changing original working flow.

There were a lot of theses kinds of thesis talking about the way to improve the structure or efficiency for proxy signature in the past but not mentioned how to make it concretely. We take XML as a basis and construct proxy signature over it successfully. It is fully compatible with standard recommendation announced by W3C organization and anything will not be changed or modified. It is really a kind of practical system and corresponding to all demands for accomplishing a proxy signature system.

Chapter 8 Future Work

The overall proxy signature system includes several parts like: proxy generation, delivery, signing, verification and so on. This thesis is focused on how to generate a proxy signature of proxy protected DSA scheme in XML form.

8.1 Proxy Signature System

In previous chapter, the XML proxy signature is feasible for implementation, but proxy generation and signature verification we didn't discuss much. Both of them should define a new structure for parameters exchange. DTD (Document Type Definition) and XML Schema is for customized use and compatible with XML related system, all parameters can be defined on demand. We didn't establish all of the subtopic to these two issues but discuss how and what we can do further.

There are some parameters needed to be generated before signing the proxy signature. The difference between conventional and proxy signature is that proxy signer should deliver random numbers like g' to original signer and original signer create (r_A, s_A) pair according to what g' is given.

On the other hand, it must be verified after signature signed. All procedures of verification are needed to be transformed to XML format too. The program of verified side has to know retrieving all needed values from XML and certificate agent if available. It needs network capability applied to program under X.509 structure.

8.2 Integration of Proxy Signature to PKI

It would be a complete solution to build up the system under PKI, and XML also provides sufficient syntax to support X.509 system establishment. The system not only supports DSA but RSA as well, so that we can pack with X.509 header for signature exchange. Here have some tags for X.509:

Table 8.2.1 XML tags for X.509

Element Name	Description
<KeyName>	A key name text-identifier.
<KeyValue>	Either an RSA or DSA public key.
<RetrievalMethod>	Allows for the remote reference of key information.
<X509Data>	X.509 certificates, names, or other related data.
<MgmtData>	Key agreement parameters (such as Diffie-Hellman parameters).

Furthermore, we can design such a system as below. Using XML parser processing the XML de-packing and dividing XML into three parts: <Signature>, <Object> and <KeyInfo> blocks. Firstly we should verify the digest value in <Signature> with transformed from <Object> block to prove its integrity. Getting the key information from not only <KeyInfo> but also <Object> block if those of three parameters stored in it.

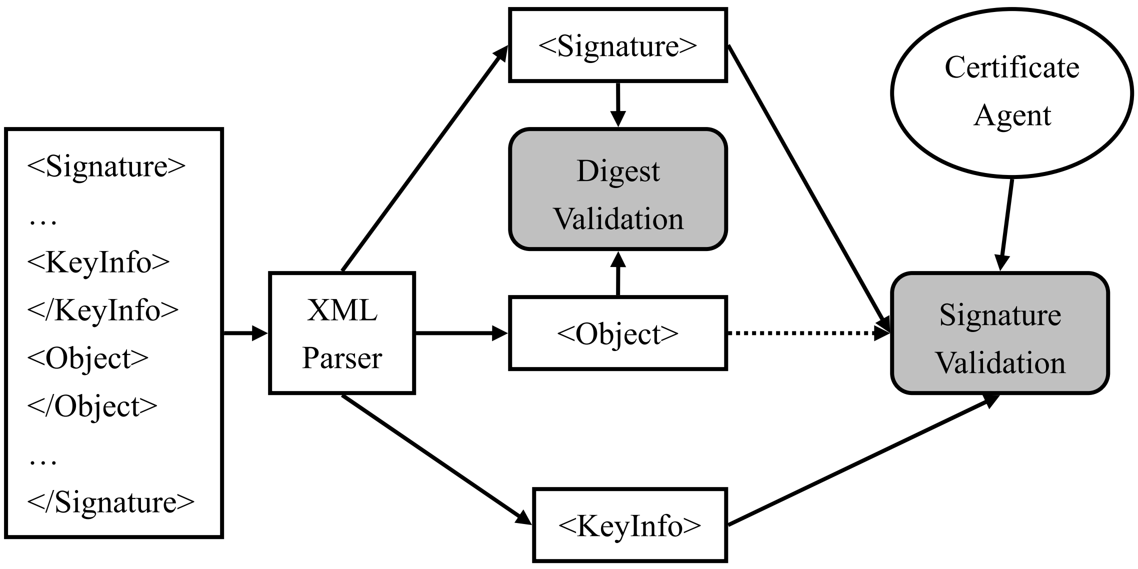


Figure 8.2.1 The procedure of PKI application

We have to emphasize that the original content instead of <Object> if using enveloped or detached proxy signature. The application needs handling all scenarios maybe occurred and extract the appropriate data out for verification. The whole process during transaction does not change anything and we can achieve our request just only modified the back-end program. There would be no pain if we apply or follow out XML proxy signature to current system.

Reference

[BD$_1$02] Blake Dournaee, "XML Security", McGraw-Hill, pp. 70-73, 2002.

[BD$_2$02] Blake Dournaee, "XML Security", McGraw-Hill, pp. 108-113, 2002.

[ElG85] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," IEEE Transactions on Information Theory, vol. 31, no. 4, pp. 469-472, July 1985.

[HWW01] Chien-Lung Hsu, Tzong-Sun Wu, Tzong-Chen Wu, "New nonrepudiable threshold proxy scheme with known signers," The Journal of System and Software 58, pp. 119-124, 2001.

[LHW98] Narn-Yih Lee, Tzonelih Hwang, Chih-Hung Wang, "On zhang's nonrepudiable proxy signature schemes," ACISP 1998, pp. 415-422, 1998.

[LK99] B. Lee, and K. Kim, "Strong proxy signatures," IEICE Trans. Fundamentals, vol. E82-A, no.1, pp.1-11, Jan 1999.

[LKK$_2$01] B. Lee, H. Kim, and K. Kim, "Secure mobile agent using strong non-designated proxy signature", Proceedings of ACISP2001, LNCS vol. 2119, Springer-Verlag, pp. 474-486, 2001.

[MUO$_1$96] M. Mambo, K. Usuda, and E. Okamoto, "Proxy signatures: delegation of the power to sign messages, " IEICE Trans. Fundamentals, vol. E79-A, no.9, pp.1338-1354, 1996.

[MUO$_2$96] M. Mambo, K. Usuda, and E. Okamoto. "Proxy signatures for delegating signing operation," In Proceedings of the 3rd ACM Conference on Computer and Communications Security (CCS), 48C57. ACM, 1996

[Neu93] B. Clifford Neuman, "Proxy-based authorization and accounting for distributed systems," International Conference on Distributed Computing Systems 1993.

[NW98] Norman Walsh, "A Technical Introduction to XML", October, 1998.

[Sch00] B. Schneier, Applied cryptography, John Wiley & Sons, 2000.

[Sun99] H. M. Sun, "An efficient nonrepudiable threshold proxy signature scheme with known signers," Computer Communications, vol. 22, no. 8, pp. 717-722, New York, IPC Science and Technology Press, 1999.

[W3C02] W3C, "XML-Signature Syntax and Processing", XML signature WG, 2002

[YC05] Yi-Shiung Yeh, Ming-Hsin Chang, "On Proxy Signatures with Forward-Secure and One-time Properties and their Applications in PKI", pp. 30-35, 2005

[Zha97] K. Zhang, "Threshold proxy signature schemes," 1997 Information Security workshop, Japan, pp. 191-199, September, 1997