Spring 2011

Analysis of Windows Cardspace Identity Management System

Thomas Hanrahan
Regis University

Regis University
College for Professional Studies Graduate Programs
Final Project/Thesis

Disclaimer

ANALYSIS OF WINDOWS CARDSPACE IDENTITY MANAGEMENT SYSTEM

A PROJECT

SUBMITTED ON 26th OF AUGUST, 2011

TO THE DEPARTMENT OF INFORMATION TECHNOLOGY OF THE SCHOOL OF

COMPUTER & INFORMATION SCIENCES

OF REGIS UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS OF MASTER OF
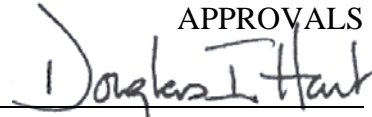
SCIENCE IN SOFTWARE ENGINEERING AND DATABASE TECHNOLOGIES

BY

Thomas Hanrahan

Thomas Hanrahan

APPROVALS

Douglas I. Hart

Douglas I. Hart

Donald J. Ina

Donald J. Ina - Ranked Faculty Name

Nancy Birkenheuer

Program Coordinator

Abstract

The Internet, which was originally developed for academic purposes, has expanded and been applied to commercial and business enterprises. It is possible to purchase airline tickets, check bank balances and communicate through e-mail with each other through the Internet. These services can all be performed relatively easily with the proliferation of Internet Service Providers and the lower cost of Personal Computers. The development of the Internet has also had a huge impact on businesses with the growth of e-commerce, e-banking and the tremendous growth in email traffic.

There is however a negative impact to this development of the Internet with the rise in on-line criminal activity. The increasing use of the Internet has resulted in the development of on-line identities for users. There can be a great deal of sensitive and personal information associated with an on-line identity and gaining access to these privileges can provide cyber criminals with access to personal resources such as bank account details, credit card information etc. .This type of activity has given rise to the term "identity theft".

This project will present an introduction to Microsoft Cardspace and how it relates to dealing with identity theft, the theory behind the application and present practical demonstrations of how the technology can be implemented using Microsoft's .NET framework technology.

Acknowledgements

Table of Contents

List of Figures

Chapter 1 – Introduction

The rise in popularity of the Internet for commercial and business environments has increased the interaction that people have with the Internet but also has had a negative impact with the resulting rise of crime related to identity theft. The Internet is now part of our daily lives, with people able to purchase items on-line, access bank accounts and perform routine tasks online. This flexibility has allowed people manage their affairs for themselves and allowed more choice for the consumer. This flexibility has also however opened up new avenues for criminals to gain illegal access to valuable data. This has manifested itself in the rise of identity theft.

What is identity theft? Bocij (2006) provides the following definition – "Identity theft (also called identity fraud) involves impersonating someone, often by using his or her personal information, such as a Social Security number, address, and credit card details. Usually identity theft is carried out with the aim of obtaining money, goods, or services at the expense of the victim."

Computer fraud has always been with us from the early days of computing when piracy was the most common cybercrime. The progression from computer piracy was the introduction of computer viruses and worms. "The idea of a computer virus is very old, but it gained real traction as potential hosts (programs) enjoyed widespread adoption and more distribution channels (BBCs, floppy disks, the first shareware). The bane of early system administrators and every dad who had fans of pirated games in the household, it elicited the creation of an entirely new software class: the antivirus applications."(Bertocci , Serack and Baker, 2007)

"If viruses weren't bad enough for shaking user's confidence in computer systems, with worms things went out of control. A worm does not need a host program. Rather, it leverages known exploits in network-enabled software for spreading from machine to machine. Email

clients, instant messaging (IM) programs, file-sharing software, even low-level network protocol implementations can be leveraged as infection vectors."(Bertocci et al., 2007)

These viruses and worms still pose a threat to modern computer systems but the advent of online services has introduced a new type valuable data to steal. These online services are not located on a local pc system but are centralized on a server. This centralization of services has placed more emphasis on the requirement of identity for accessing online services. When accessing a service online it is difficult to identity the source of the web page unlike in real life eg. if a customer requires to visit the bank, it is easy to identify the bank building as they walk down the street allowing them to enter the bank and carry out the required transactions. This is not possible online and a level of trust has to be placed on the service provider and the current webpage displayed in order to use the required service.

The accessing of services online and implementation of transactions raises identity related issues such as authentication and authorization. These issues have been discussed at length over the years and this discussion has led to the development of the seven laws of identity and the identity metasystem. These concepts will be discussed in Chapter 2 and how they apply to the Windows Cardspace system will be presented.

The proliferation of online services has led to the rise in password fatigue – all different systems have different logon requirements – different password requirements and logon name requirements – it has become difficult for users to remember the logons and passwords for each service that they require to logon to. A significant effort has gone into developing federated logon systems to provide SSO (single sign on) capability. In an attempt to remove password fatigue and thus simplify identity management a number of identity management systems have

been developed such as the Kantara Initiative (formally Liberty Alliance), Open ID, Microsoft

Cardspace and the latest product from Microsoft U-Prove.

The subject of this project is the Windows Cardspace system introducing the concepts of

the system, some background information in relation to the Identity Metasystem and an

introduction to the development requirements using the .Net Framework and Visual Studio 2010.

Microsoft's latest technology U-Prove will be presented with an overview of the current SDK.

Chapter 2 – Review of Literature and Research

This chapter introduces the research conducted into the subject of Cardspace and presents the background information and development tools required for the development of applications which incorporate Windows Cardspace.

Digital Identity System

Who are you? As we go about our every day business and travel through the world, different forms of identity will be required. If we are travelling by air across continents we will have to validate our identity in the form of a passport. If stopped by a policeman, a driver's license may be required to provide proof of identity, proof of age may be required to purchase alcohol. These forms of identity are well established and there exists a chain of trust. A passport is issued by a recognized authority which in turn requires valid forms of identity in order to issue a passport. This valid identity required by the passport issuing authority will be a birth cert which has been issued by a recgonised authority, thus establishing a chain of trust. In the online world, identity validation also exists through the use of passwords and usernames – as with the real world, digital identities can exist in different formats eg a logon for an email account, a secure form of logon for a bank account or a network logon at work.

"All of these contexts have well-understood ways for you to establish your identity. Yet, in one very important context—the networked world—identity is currently a much more muddled thing. Just as in the physical world, all of us have a variety of *digital identities*, and they're expressed in different ways. Today, however, there's no consistent way to deal with this portfolio of digital identities. Instead, we're left struggling in a complex, confusing, and insecure environment. Yet different kinds of digital identities will always be necessary—no single

identity will suffice. And the reality is that these identities will always be provided by a range of different sources—no single identity provider will suffice, either. This means that the solution is not to mandate a single system for digital identity, but rather to find a coherent way to use multiple digital identity systems. What's required is a system of systems—a *metasystem*—focused on identity." (Chappell, 2006)

. A digital identity can be defined as "a set of claims made by one digital subject about itself or another digital subject". (Cameron, 2005) Cameron also defines a claim as "an assertion of the truth of something, typically one which is disputed or in doubt." The following are examples of claims in the digital world:

- A claim could just convey an identifier—for example, that the subject's student number is 490-525, or that the subject's Windows name is REDMOND\kcameron. This is the way many existing identity systems work.

- Another claim might assert that a subject knows a given key—and should be able to demonstrate this fact.

- A set of claims might convey personally identifying information—name, address, date of birth and citizenship, for example.

- A claim might simply propose that a subject is part of a certain group—for example, that she has an age less than 16.

- And a claim might state that a subject has a certain capability—for example, to place orders up to a certain limit, or modify a given file.

Laws of Identity

The "Laws of Identity" (Cameron, 2005) are intended to codify a set of fundamental principles to which any universally adopted, sustainable identity architecture must conform. The Laws were proposed, debated, and refined through a long-running, open, and continuing dialogue on the Internet. Taken together, the Laws define the architecture of the identity metasystem.

The Laws of Identity are defined as follows:

Law # 1 User Control and Consent

This law relates to a technical identity system revealing information identifying a user only with the user's consent.

This is a fundamental principle for any identity system and puts the user in control. The technical identity system must provide several items to help make sure the decision to consent is an informed one. According to Mercuri (2007), prior to identifying what *claims* are being requested, the user must be first told who is requesting them. A user must be presented with the opportunity to check the certificate of the site or service requesting it and also have the ability to examine the privacy policy.

According to Mercuri (2007), the user must have the opportunity to consent and be in control of who receives their information and what information the relying party will receive. The ability to inspect a display token for a managed card and see what information a third party is releasing about the user must also be available.

Law # 2 Minimal Disclosure for a Constrained Use

The solution that discloses the least amount of identifying information and best limits its use is the most stable long-term solution.

Why provide more information than is required - "We should build systems that employ identifying information on the basis that a breach is always possible. Such a breach represents a risk. To mitigate risk, it is best to acquire information only on a "need to know" basis, and to retain it only on a "need to retain" basis." (Cameron, 2005)

Law # 3 Justifiable Parties

Digital identity systems must be designed so the disclosure of identifying information is limited to parties having a necessary and justifiable place in a given identity relationship.

The identity system must make its user aware of the party or parties with whom she is interacting while sharing information. "If you were buying a book or DVD on Amazon, what justifiable role did Microsoft have in that interaction? – None. If you were logging in to check your bank account balance online, what justifiable role did Microsoft have in that interaction? None." (Mercuri, 2007)

Law # 4 Directed Identity

A universal identity system must support both "omnidirectional" identifiers for use by public entities and "unidirectional" identifiers for use by private entities, thus facilitating discovery while preventing unnecessary release of correlation handles.

The definition of Omnidirectional and Unidirectional is provided by Mercuri (2007):

Omnidirectional : If you do online banking or shop online today, you'll notice that the transactions are done using Secure Sockets Layer (SSL). The data between the site and your computer is encrypted. This is done through the use of a certificate. The certificate is attached to a website, and it broadcasts its identity to everyone. You can see who the certificate was issued to, who it was issued by, and the dates the certificate is valid. Here the identity of the website is omnidirectional. The identity is broadcast in public to anyone who uses the site.

Unidirectional : These are identifiers used between private entities. If you look at your bank statement or your credit card statements, you'll see that you have unique identifiers associated with you for the institution in the form of an account number. If you have multiple accounts, you have unique numbers for each of the accounts. These are unidirectional identifiers.

This law relates to no two relying parties sharing identity information. A universal identity system may have a single card which can be used across multiple locations, but for each location, a unique site-specific identifier is created. While using the same information card across multiple sites, each site—at the relying party—would see a different identifier for that card

Law # 5 Pluralism of Operators and Technologies

A universal identity system must channel and enable the interworking of multiple identity technologies run by multiple identity providers.

There must be a pluralism of technologies in order accommodate multiple different solutions. When it comes to digital identity, it is not only a matter of having identity providers run by *different parties* (including individuals themselves), but of having identity systems that offer *different (and potentially contradictory) features.*

"If you look at the physical world, government-issued passports are quite different. Each country issues its own passports that are good both for identification within their borders and for entry into and identification in foreign countries. The countries that issue passports follow certain standards in the issuance of their passports, and the documents are accepted in most places in the globe." (Mercuri, 2007)

Law # 6 Human Integration

The universal identity metasystem must define the human user to be a component of the distributed system integrated through unambiguous human-machine communication mechanisms offering protection against identity attacks.

"Although much can be done to secure the connections between identity providers, relying parties, and the desktop of the consumer, there is still the challenge of the two feet between the desktop and the user." (Mercuri, 2007)

"A number of things can happen in those two feet that can defeat the most expensive and complex software and hardware security. When defining an identity solution, one must recognize that even with the most expensive and efficient technology in the world, if the aspect of how humans interact with that technology is not properly thought out, it leaves the door open for breaches of security as a result". (Mercuri, 2007)

Law # 7 Consistent Experience Across Contexts

The unifying identity metasystem must guarantee its users a simple, consistent experience while enabling separation of contexts through multiple operators and technologies.

"Regardless of the underlying protocols, technologies, and operators, there needs to be a level of consistency, a set of expectations for how identity requests should appear, how the interactions should take place and what one can expect to occur. These expectations need to be consistent and expected." (Mercuri, 2007)

Identity Metasystem

The identity metasystem is born is out of law #5 *Pluralism of Operators and Technologies* – diversity is part of the Internet ecology. In order for an identity system to comply with this law, a solution has been proposed to define a system of systems ie. a metasystem. A Metasystem abstracts away concepts that are common to all identity systems but which are often implemented differently.

"This metasystem, or system of systems, would leverage the strengths of its constituent identity systems, provide interoperability between them, and enable creation of a consistent and straightforward user interface to them all. The resulting improvements in cyberspace would benefit everyone, making the Internet a safer place with the potential to boost e-commerce, combat phishing, and solve other digital identity challenges. The identity metasystem makes it easier for users to stay safe and in control when accessing resources on the Internet. It lets users select from among a portfolio of their digital identities and use them at Internet services of their choice where they are accepted. The metasystem enables identities provided by one identity system technology to be used within systems based on different technologies, provided an intermediary exists that understands both technologies and is willing and trusted to do the needed translations. An intermediary exists that understands both technologies and is willing and trusted to do the needed translations."(Microsoft, 2005)

The Identity Metasystem has three roles– these being the

- Identity Providers who will issue digital identity.

- Relying Parties who require the identity eg. a website or online service that utilizes identities offered by other parties.

- Users – the individuals who the claims are made about.

The above are the roles that compromise the metasystem – the metasystem also has five key components which are defined as:

- Claims Based Identities

- Negotiation

- Encapsulating Protocol

- Claims Transformation

- Consistent User Experience

Claims Based Identities

Claims Based Identities are at the core of the Metasystem and as presented already, claims are statements about an entity that an identity provider asserts are valid. "Although the most common claims will be from the pool used in the self-issued cards, a claim can be anything. A claim can be specific to an identity provider—for example, a membership number, bank account number, and so on. Relying parties provide statements of requirements expressed in terms of WS-Policy".(Mercuri, 2007) WS- Policy will be presented later in this chapter.

Negotiation

Negotiation enables participants in the metasystem to make agreements needed for them to connect with one another within the metasystem. In the metasystem how will a relying party negotiate to retrieve the information it requires. "Relying parties specify the claims they are interested in and whether those claims are required or optional. They also specify the specific type of tokens (X.509, Kerberos, and so on) that they are willing to accept. Negotiations are conducted using WS-MetadataExchange and WS-SecurityPolicy." (Mercuri, 2007)

If one party understands SAML and X.509 claims, and another understands Kerberos and X.509 claims, the parties would negotiate and decide to use X.509 claims with one another.

Another type of negotiation determines whether the claims needed by a relying party can be supplied by a particular identity. Both kinds of negotiation are simple matching exercises; they compare what one party can provide with what the other one needs to determine whether there's a fit.

Encapsulating Protocol

In order for negotiation to take place data must flow between the relevant parties. Encapsulating protocol provides a technology-neutral way to exchange claims and requirements between subjects, identity providers, and relying parties. Participants determine the content and meaning of what is exchanged. If two parties agree to use SAML for their transaction –eg the encapsulating protocol would allow an application to retrieve SAML-encoded claims without having to understand or implement the SAML protocol.

Claims Transformers

Claims transformers bridge organizational and technical boundaries by translating claims understood in one system into claims understood and trusted by another system. "Despite the best intentions, not everyone will use the same grammar to define their claims. In other cases, it might not be desirable or necessary to provide the claim itself; instead, it is something that is established by the claim. In these cases, you would require claims transformers. These would take claims as presented on an information card and transform them to a claim that is understood by the relying party". (Mercuri, 2007) The encapsulating protocol used for claims transformation is WS-Trust.

An example of a claims transformation would be where the claim "Born on March 22, 1960" could be transformed into the claim "Age is over 21 years", which intentionally supplies less information and would comply with the identity Law #2 relating to minimal disclosure for a constrained use.

Consistent User Experience

The identity system should present a consistent interface to the user and endeavor to make important information obvious. The identity metasystem, therefore, seeks to empower users to make informed and reasonable identity decisions by enabling the development of a consistent, comprehensible, and integrated user interface for making those choices.

WS-* Specifications

"Cardspace leverages an open set of XML-based protocols. These include WS-Trust, WS-MetadataExchange and WS-SecurityPoilcy. As a direct result any technology or platform that supports WS-* protocols can integrate with Cardspace". (Liberty & Horovitz, 2008) The Identity Metasystem requires a platform agnostic implementation – early solutions were COM+, Java EJB and COBRA.

The emergence of HTML and XML was to alter this scene due in no small measure to the one major advantage of HTML ie. its minimal requirements, resilience to errors and interpretations. Following on from HTML and XML came SOAP (Simple Object Access Protocol) – SOAP is a protocol for message exchange. Within the web services environment, according to Britton & Bye (2004), SOAP is the key messaging technology. It is relatively simple and can be used over many communication protocols. SOAP has the advantage of widespread support as a standard and is consequently widely implemented. Security is an issue with SOAP as no method is provided to implement security. SOAP has been intentionally kept simple and in order to address this situation, the industry introduced additional web service specifications.

The specifications were named according to a consistent pattern: WS-Security describes how to add security capabilities to SOAP messages, WS-ReliableMessaging establishes a protocol for adding reliability assurances to web services communications, and so on. That earned them the collective name of the WS-* specifications.

Web Services

The definition offered by the W3C of a web service is: "*A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards*".*(Web Services Architecture, 2004)*
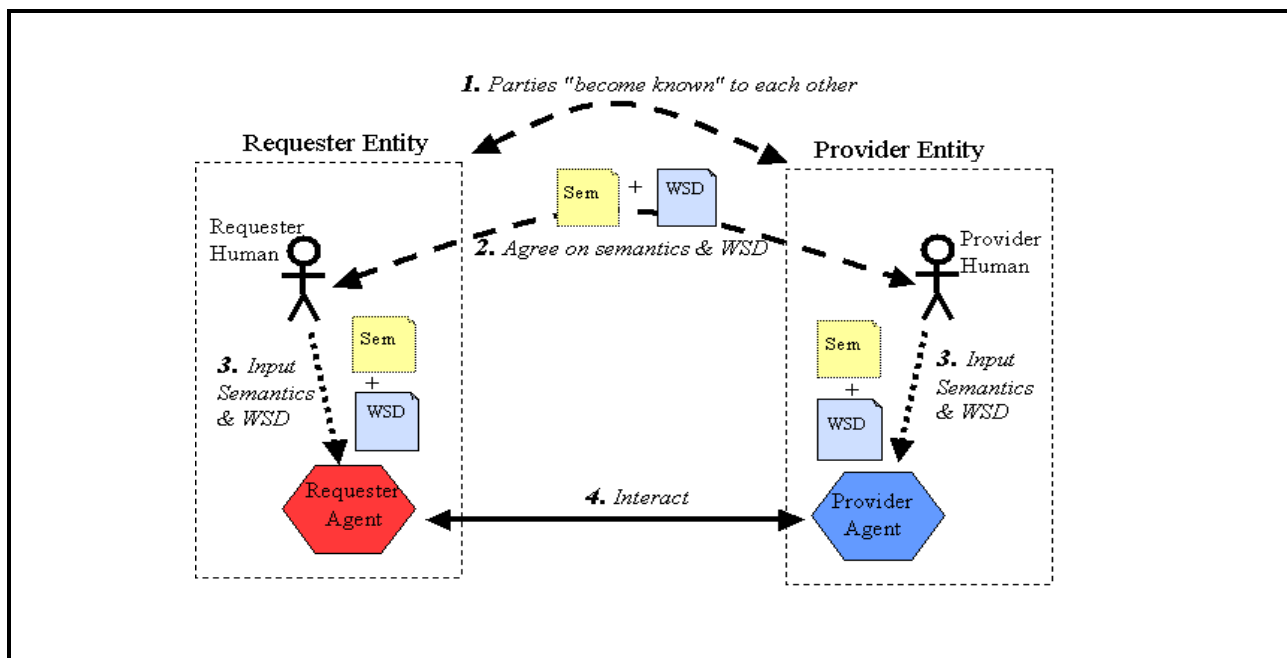
Figure 1 Web Service Basic Architectural Roles (Web Services Architecture, 2004)

Figure 1 presents an overview of the Web service process. "The agent is the concrete piece of software or hardware that sends and receives messages, while the service is the resource characterized by the abstract set of functionality that is provided. To illustrate this distinction, you might implement a particular Web service using one agent one day (perhaps written in one programming language), and a different agent the next day (perhaps written in a different programming language) with the same functionality. Although the agent may have changed, the Web service remains the same." (Web Services Architecture, 2004)

The implementation of an identity metasystem will require a methodology where according to Bertocci et al. (2007), the identity layer requires supplying a concrete, interoperable implementation of the components encountered already: a claim-based identity representation, a negotiation protocol, an encapsulation protocol, and so on. Those components must guarantee state-of the-art security at every stage, but they must be technology and platform-agnostic. The technology that satisfies these requirements is known as Web Services.

"Web services solved many of the challenges related to the attempt to put into practice the principles behind the Identity Metasystem. They are platform-agnostic by design, they pay special attention to security, and they are a technology widely available in the product offerings of the main IT vendors and in the Open Source world." (Bertocci et al., 2007)

Web Services allow for a way of defining how to communicate with your software. This is achieved through the use of XML – (Extensible Markup Language) which is an immensely popular markup language, which has the advantage of being readable cross-platform and, when the complexity and size permits it, by humans as well. As already stated, SOAP is the messaging protocol implemented - SOAP defines what a web service message should look like and how it can be sent between two endpoints. SOAP represents everything using XML and SOAP is extensible by design.

Web Services are comprised of the following areas:

- WSDL – Web Services Description Language
- WS-Addressing
- WS-Policy
- WS-Security

WSDL – Web Services Description Language

The definition of a web services states that has an interface described in a machine-processable format (specifically WSDL). WSDL describes the messages a web service accepts and produces – "for an application to use a Web service, the programmatic interface of the Web service must be precisely described."(Ryman, 2003)

Figure 2 presents a sample of a WSDL file generated for a stock Quote Service – the WSDL defines the types for GetLastTradePrice and GetLastTradePriceResponse . "The remainder of the file builds up the Web service definition by assembling the types into messages, the messages into operations, and the operations into port types. It then binds the port type to the SOAP/HTTP protocol, and finally assigns it the address http://www.stockquoteserver.com." (Ryman, 2003)

WS-Addressing

A typical URL for the Internet such as http://www.homepage.com/home.htm specifies a number of items such as the protocol to be applied for this communication ie. http. The location of the document home.htm is also specified – it is located in the path /home.htm on the server www.homepage.com.

"The same mechanism could be used for web services, and in fact this is common practice in many applications. However, this does not play very well with web services' attempts to be independent from the underlying technology. HTTP mandates the use of one specific protocol, whereas the web service should be able to be moved on some other transport without dependencies. WS-Addressing provides a richer way of referring to web services, helping to

overcome the previously mentioned limitations and supplying the more expressive model that is required by the other advanced WS-* specifications." (Bertocci et al., 2007)

WS-Policy

In terms of a web service WSDL describes what the web service offers and the format of the messages required – but according to Bertocci et al. (2007) it doesn't give any other details required by the web service. They offer the example of a web service implementing a wire transfer which may be invoked with the correct message, but the software will not execute the operation unless the caller identifies itself using a certain authentication technique.

"Service metadata can describe the capabilities and requirements of a Web service, i.e. representing whether and how a message must be secured, whether and how a message must be delivered reliably, whether a message must flow a transaction, etc. Exposing this class of metadata about the capabilities and requirements of a Web service enables tools to generate code modules for engaging these behaviors. Tools can use this metadata to check the compatibility of requestors and providers. Web Services Policy can be used to represent the capabilities and requirements of a Web service." (Vedamuthu & Roth , 2006)

"Web Services Policy is a machine-readable language for representing the capabilities and requirements of a Web service. These are called 'policies'. Web Services Policy offers mechanisms to represent consistent combinations of capabilities and requirements, to determine the compatibility of policies, to name and reference policies and to associate policies with Web service metadata constructs such as service, endpoint and operation. Web Services Policy is a simple language that has four elements - Policy, All, ExactlyOne and PolicyReference - and one attribute - wsp:Optional."(Vedamuthu & Roth , 2006)

WS-Security

WS-Security builds on the extensible capabilities of SOAP by defining ways of protecting SOAP messages and providing means of transporting security-related information. The rise in popularity of XML has meant that XML has been used for message transmissions but these files are readable by humans and thus do not provide adequate security for security conscious applications. WS-Security is the solution to solve this problem.

"WS-Security defines a SOAP Header into which message senders can insert security information such as digital signatures, encrypted data, and security tokens. WS-Security defines a security token as a collection of one or more claims, typically made by the message sender and often digitally signed by some trusted authority so that the message recipient will have reason to trust those claims. WS-Security supports both binary security tokens (e.g. X.509 certificates and Kerberos tickets) as well as XML-based tokens – importantly, including SAML assertions." (Madsen, 2004)

This introduces the topic of SAML (Security Assertion Markup Language) – SAML was launched in 2002 by the OASIS (Organization for Advancement of Structured Information Science). SAML is an XML- based framework for communicating user authentication, entitlement and attribute information.

Figure 2 illustrates the high-level structure of a SAML authentication assertion. "The outer structure of an assertion is generic, providing information that is common to all of the statements within it. Within an assertion, a series of inner elements describe the authentication, attribute, authorization decision, or user defined statements containing the specifics." (Madsen, 2004)
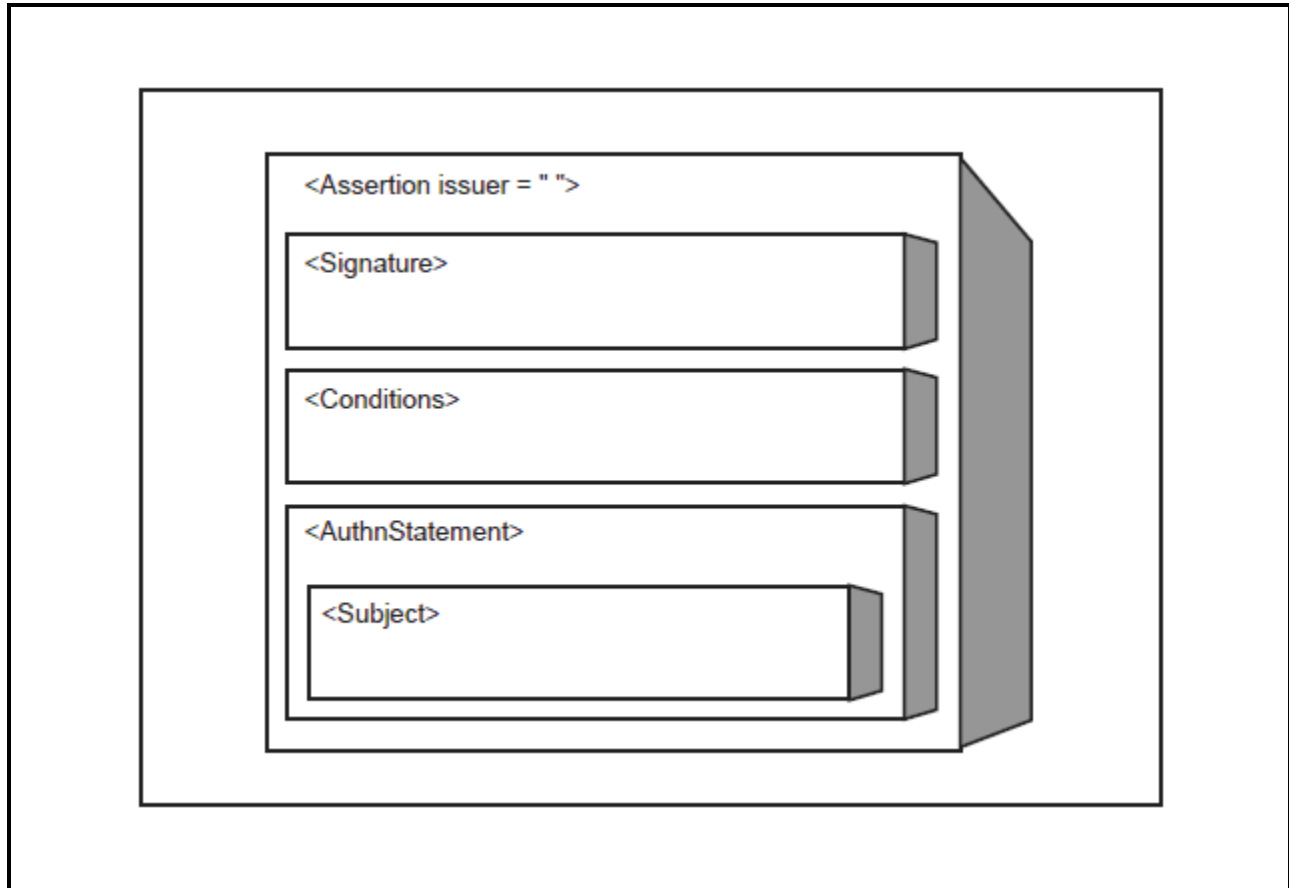
Figure 2 SAML Authentication Statement (Madsen, 2004)

SAML has emerged as the gold standard for federated identity. Identity federation benefits both users and enterprises - providing principals with a smooth, cross-domain browsing experience through single sign-on (SSO). SSO can be defined as the capability of accessing multiple resources that require authentication while requiring the user to go through the authentication experience only once.

Web Service Security Mechanisms

The security mechanisms that have been developed for SOAP are XML Signature and XML Encryption. According to the XML-signature Requirements working draft document, the

XML 1.0 Recommendation describes the syntax of a class of data objects called XML documents. The mission of this working group is to develop a XML syntax used for representing signatures on digital content and procedures for computing and verifying such signatures. Signatures will provide data integrity, authentication, and/or non-repudiatability.

XML Signature

A signature is defined as a value generated from the application of a private key to a message via a cryptographic algorithm such that it has the properties of integrity, message authentication and/or signer authentication. XML Signatures are applied to arbitrary digital content (data objects) via an indirection. These data objects are digested and the resulting value is placed in an element (with other information).The element is then digested and cryptographically signed. XML digital signatures are represented by the Signature element within the XML document and data objects are the actual/octet data being operated on by an application.

According to Lakshminarayanan (2008) the XML signature schema is a W3C standard that describes how the digital signature information is represented in an XML document. XML signature schema can not only describe the digital signature of an XML element or XML element content, but also an image or an externally-referenced entity. In any case, the recipient of the XML signature message has to know:

- Data that is being signed.

- Digest value of that data.

- Digest algorithm used.

- Digital signature information, such as signature value, key information.

The digitally-signed message that adheres to the XML signature schema always starts

with a signature XML element which contains the following main child elements:

- SignedInfo

- SignatureValue

- KeyInfo

- Object

The mentioned XML elements can have their own child elements that can specify additional

instructions to process the digitally-signed message.
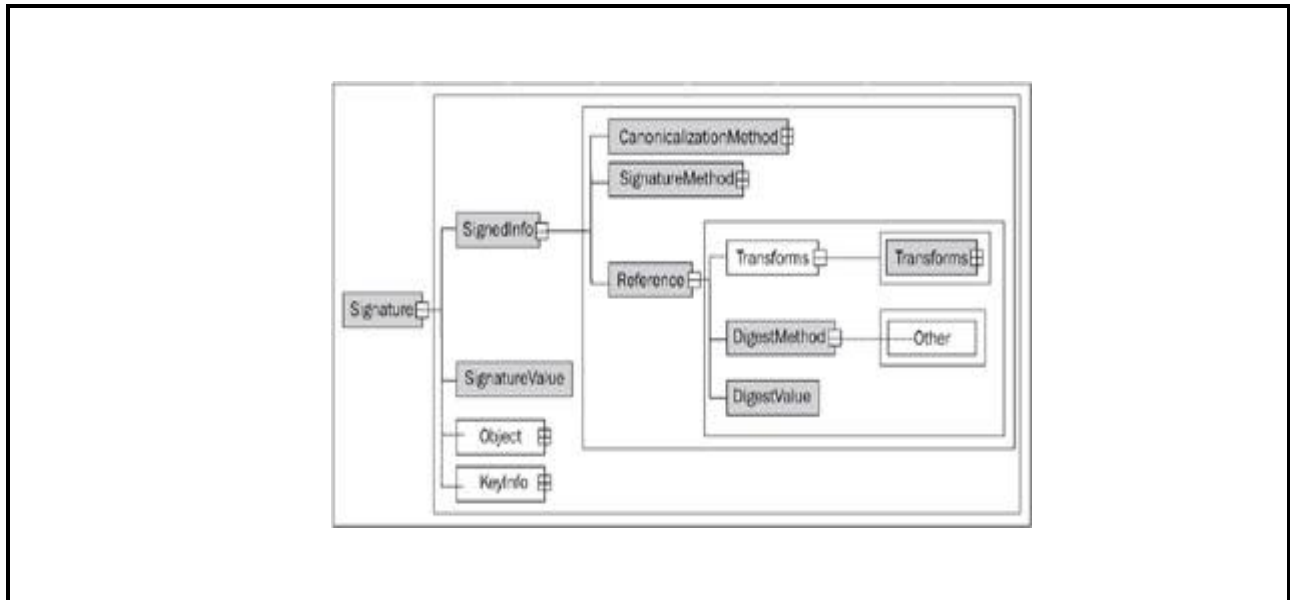
Figure 3 Overview of XML Signature Schema (Lakshminarayanan, 2008).

Figure 4 shows an overview of the XML signature schema with the signature element as

the root element and its child elements. The figure also shows the order in which those elements

appear, which is very important, as described in the XML signature schema.

XML Encryption

Confidentiality in web services is ensured by encrypting the confidential data in the web service messages. Encryption is nothing but a process of applying an algorithm using an encryption key over the plain text to create the cipher text (encrypted data). The requirements for XML Encryption are that it encrypts only portions of the XML that needs to be encrypted. The service provider and consumer should not be tied to the same set of programs to encrypt and decrypt. In order for anyone who has the appropriate keys to decrypt the data, the encrypted data should be represented in such a way that the application can determine:

- Where the encrypted data is placed inside the XML message.

- What encryption algorithm is used to encrypt the data.

- Any other cryptographic parameters.

When an XML element content is encrypted, the encrypted data element replaces the element or content (respectively) in the encrypted version of the XML document. By definition, the XML encryption schema is nothing but an XML schema that outlines how the encrypted data can be described in an XML format. Any application that can understand the XML encryption schema should be able to encrypt and display data as per the schema, and should also be able to decrypt the data. Figure 4 presents an overview of an example of XML encryption.

The relevant elements presented in this example are presented as follows:

- EncryptedData – this element represents all the data.

- CipherValue – this element represents the encrypted data.

- EncryptionMethod – this element represents the algorithm used, in the example presented in Figure 5 the algorithm specified is Triple DES.

- KeyInfo – this element describes the symmetric key information.

- CipherData – this element represents the cipher text or encrypted text.

```
    <xenc:EncryptedData xmlns="http://www.w3.org/2001/04/xmlenc#"
xmlns:xenc="http://www.w3.org/2001/04/xmlenc#" Type="http://www.
w3.org/2001/04/xmlenc#Element" Id="ED">
        <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/
xmlenc#tripledes-cbc"/>
        <dsig:KeyInfo xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
          <dsig:KeyName>EncryptedElementWithSymmKey</dsig:KeyName>
        </dsig:KeyInfo>
        <xenc:CipherData>          <xenc:CipherValue>nrNckdqE5O1CCZkjAORM9
s7mOYv8t4QUJ/tE1ONdfUe8zWaXeJuDYAl5dQ4Ypfpu</xenc:CipherValue>
        </xenc:CipherData>
    </xenc:EncryptedData>
```

Figure 4 Overview of XML Encryption (Lakshminarayanan, 2008).

WS-Trust

According to Bertocci et al, (2007), WS-Trust extends WS-Security with methods for

issuing, renewing, and validating security tokens in a platform-agnostic manner. WS-Trust

introduces a special kind of web service, called Security Token Service (STS). STS transforms

WS-Security tokens - one token enters another token exits.

WS-MetadataExchange

According to Bertocci et al, (2007), web services use metadata to describe what other

endpoints need to know to interact with them. As part of Web Services, WS-Policy describes the

capabilities, requirements, and general characteristics of Web services. WSDL describes abstract

message operations, concrete network protocols, and endpoint addresses used by Web services.

Web service–based transactions grew in complexity resulting in the need to define how

to acquire service metadata in a standard and programmatic fashion WS-MetadataExchange

allows one caller to query one web service and obtain its metadata information, typically

WSDL/policies.

WS-SecurityPolicy

"WS-SecurityPolicy defines an assertion framework which contains a collection of

assertions and assertion operators aimed at expressing security requirements for the invocation of

web services. WS-SecurityPolicy builds upon the more generic WS-Policy standardizing how to

express requirements such as how to mandate in a message the presence of a security token of a

certain shape, which parts of a message should be signed or encrypted and with which keys, and

so on."(Bertocci et al., 2007)

WS-Federation

A federation is a collection of realms (security domains) that have established

relationships for securely sharing resources eg a resource provider in one realm can provide

authorized access to a resource it manages. Authorization is based on claims about a principal

(such as identity or other distinguishing attributes) that are asserted by an Identity Provider (or

any Security Token Service) in another realm.

The fundamental goal of WS-Federation is to simplify the development of federated

services through cross-realm communication and management of Federation Services by re-

using the WS-Trust Security Token Service model and protocol.

WS-Federation does not restrict users to a specific security token format. Instead, WS-

Federation builds on the WS-Trust encapsulation mechanism, the RST/RSTR, which allows

protocol processing to remain agnostic of the type of token being transmitted. This enhances the

interoperability and migration of customer deployed products as the industry introduces new and

better security token formats.

Building on the WS-Trust foundation the WS-Federation protocol provides mechanisms

that simplify interactions between the participants. A well documented method for exchanging

federation metadata makes it easy to bootstrap trust relationships and also to determine policies

for obtaining services. Cross organizational identity mapping and distributed sign-out improve

the utility and overall security of accessing federated service providers by minimizing the user's

need to manage many identifiers and tokens.

WS-* Implementation Example

Bertocci et al, (2007), present an example outlining the implementation details for

Cardspace and the WS-* standard applied to the relevant transaction. The following figure

presents an overview of the messages involved in the web services application of Cardspace.

The figure depicts the interaction among the three roles of the identity metasystem in the

canonical scenario showing which WS-* standards are used for implementing every step

S   = Subject

RP = Relying Party

IP = Identity Provider

S wants to use RP, which in turn requires its callers to present an identity issued from the

IP to authorize access. S is the buyer, RP is the seller, and IP is whatever institution issued an

identification document to the buyer, and Claim1 or Claim2 is for example an age claim.

Step 1 - S wants to call RP. The S's agent reaches out to the RP via WS-

MetadataExchange, to acquire the RP's policy and requirements. The WS-

MetadataExchange returns a WS-Policy document containing some WS-

SecurityPolicy assertions. The RP states
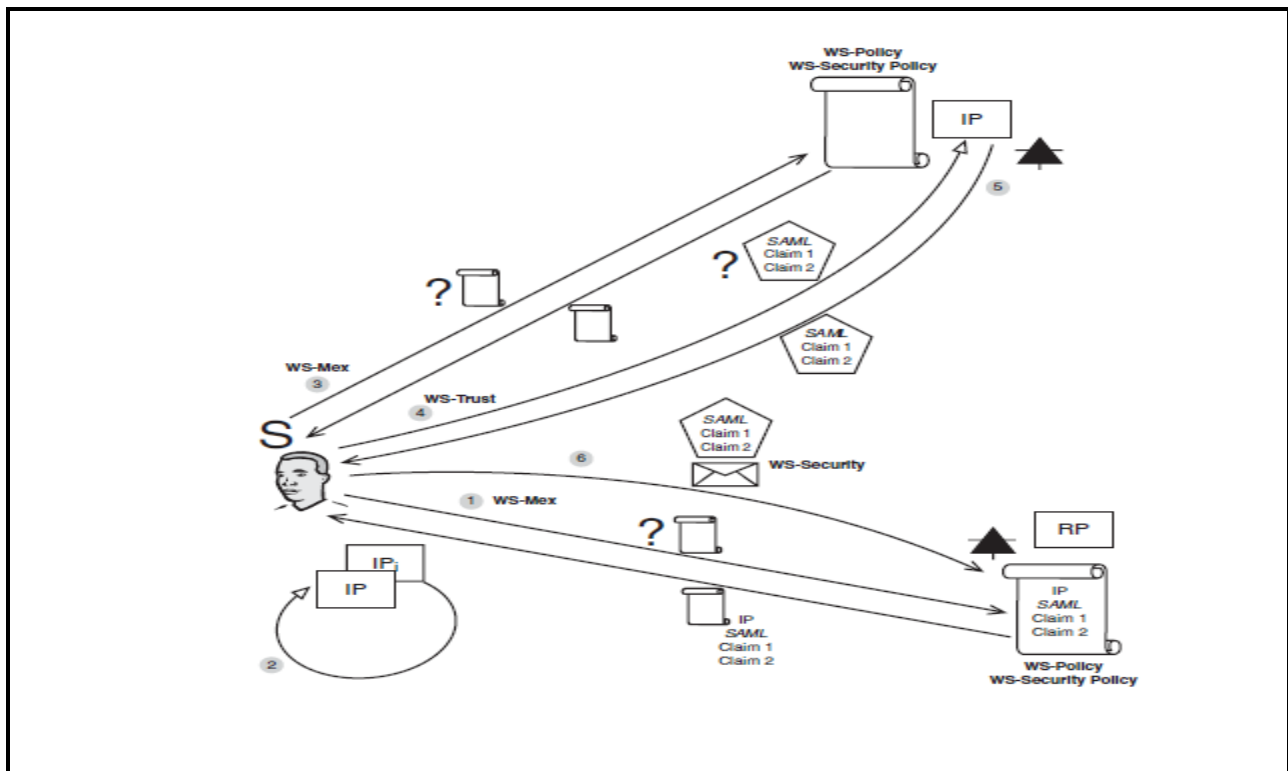
Figure 5 WS-* Application Example (Bertocci et al., 2007).

that it will consider for authentication only the users presenting an identity issued by

IP's STS, in SAML1.1 format and containing Claim1 and Claim2.

Step 2 -The S's agent checks if S has a relationship with IP that would allow it to ask for

a token of the right format and with the requested claims in it. It then presents to S its options

(that is, all the courses of actions that will end with the acquisition of a token satisfying RP's

policy).

Step 3 - Assuming that S does have a suitable relationship with IP and that S chooses to

pursue that option among the ones offered by the agent, S's agent uses WSMetadataExchange

for acquiring IP's invocation policy.

Step 4 - The S agent uses the information acquired in the former step for requesting an identity from IP's STS, by sending an appropriate RST. The agent will also take care of finding the token that the IP-STS requested for securing the RST

Step 5 - The S's agent receives the RSTR from IP, and with it the required token. The S's agent returns the token to S. S goes through the experience of examining the details of the identity, such as the content of Claim1 and Claim2, and decides whether it consents to the disclosure of that information to RP

Step 6 - If S decides to disclose, it uses WS-Security for securing the token obtained from IP the invocation to RP

Cardspace Implementation

There are two possible implementations for Cardspace – Cardspace may be implemented as a service or as a web site. In the case of a service it can be implemented as a dedicated rich client accessing a service or through a client browser accessing a web site as the relying party. The full implementation details for these scenarios will be presented in Chapter 4.

Cardspace Retired

On February 15, 2011 Microsoft announced that they would not be shipping Cardspace 2.0 –in relation to Cardspace, Microsoft claim that:

- Windows Cardspace was initially released and developed before the pervasive use of online identities across multiple services.

- The identity landscape has changed with the evolution of tools and cloud services.

- Microsoft claim that claims-based identity remains a central concept for Microsoft's identity strategy with the release of U-Prove

U-Prove takes the form of a user agent that takes account of cloud computing realities and takes advantage of the high-end security and privacy capabilities within the extended U-Prove cryptographic technology.

The background to U-Prove begins with a company called Credentica, (http://www.credentica.com/) which developed the U-Prove technology and was acquired by Microsoft in March, 2008. "The U-Prove technology is an advanced cryptographic technology that, combined with existing standards-based identity solutions, overcomes the long-standing dilemma between identity assurance and privacy". (U-Prove Community Technology Preview R2, 2011)

"This solution proposes to unlock a broad range of scenarios that have historically been out of the reach of both the private and public sectors - cases where both verified identity information and privacy are required. At the core of Microsoft's vision are **U-Prove Agents**—software that acts as an intermediary between websites and allows users to share their personal information in a way that helps protect their privacy".(U-Prove Community Technology Preview R2, 2011)

U-Prove will allow users to cryptographically protect data wherever it may travel. Organisations can share digital data through the users the data pertains to or through brokers or outsourcing suppliers. Intermediate parties can view the data that is to be shared allowing them to boycott inappropriate exchanges ie. selective disclosure is a feature of U-Prove – just the required data needed be shared.

A system overview of U-Prove according to Brown, Stradling and Wittenberg (2011) is presented in Figure 7. A U-Prove Agent that acts as an intermediary, aligned with the user between the issuer of identity information and the services that consume it. The goal is to enable the exchange of verified identity information from sources (Claims Provider on the left), under the user's control (via the U-Prove Agent), to the recipients (Relying Party on the right).

U-Prove Technology

"U-Prove is an innovative cryptographic technology that enables the issuance and presentation of cryptographically protected claims in a manner that provides multi-party security; issuing organizations, users, and relying parties can protect themselves not just against outsider attacks but also against attacks originating from each other. At the same time, the U-Prove technology enables any desired degree of privacy (including authenticated anonymity and pseudonymity) without contravening multi-party security. These user-centric aspects make the U-Prove technology ideally suited to create the digital equivalent of paper-based credentials and the plastic cards in one's wallet." (Brown, Stradling and Wittenberg, 2011)

Figure 6 presents a high level overview of the parties involved in the U-Prove transaction – it shows the user as being at the centre and in control of the identity information being transmitted to the relying party/service.

"The U-Prove technology is based on the protocols designed by Dr Stefan Brands. The MIT Press published a book that contains the full details and mathematical analysis of these protocols. In the seven years prior to this publication, the protocols were widely published in leading cryptography and IT security journals and proceedings. During this time the protocols
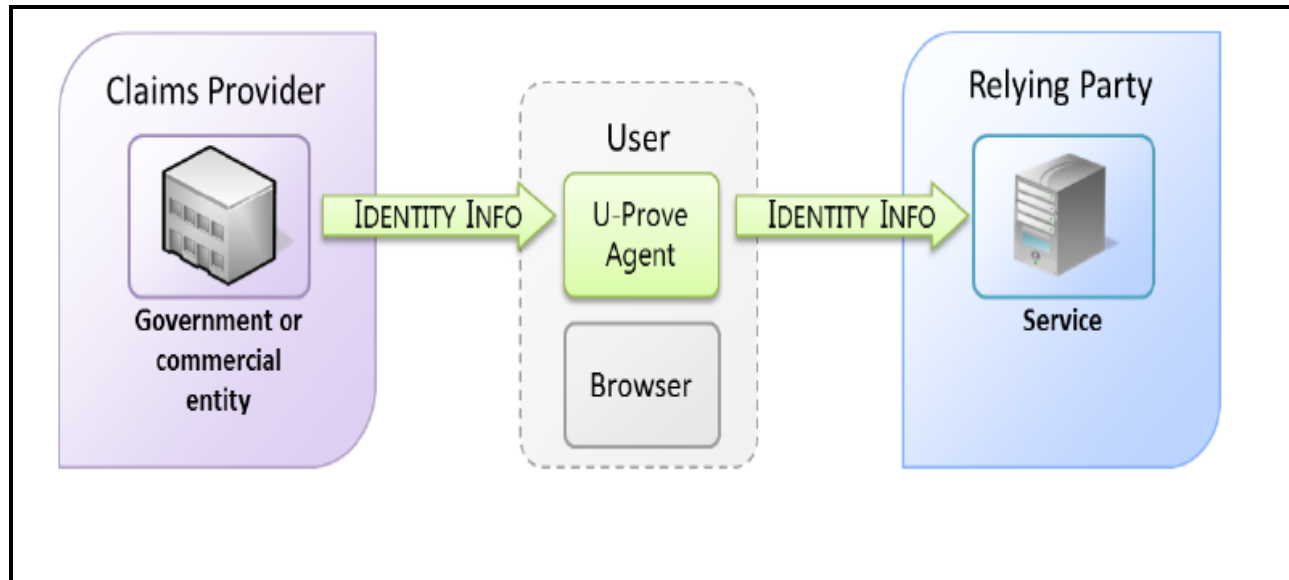
Figure 6 U-Prove System Overview (Brown, Stradling & Wittenberg 2011).

were also extensively scrutinized by dozens of professional cryptographers. Furthermore, the cryptographic protocols have been or are being taught in advanced cryptography courses at leading universities around the world". (Brown et al., 2011)

U-Prove Information Card

A U-Prove information card is provisioned like any other managed card. Figure 7 illustrates the flow for information card interactions. According to Figure 7, "when the card is used, the Identity Selector (the requester) creates a presentation token sent to a Relying Party using one or more U-Prove tokens obtained from the Identity Provider. Owing to the properties of the U-Prove technology, multiple UPTs certifying the card's claim values MAY be obtained in advance (e.g., when the card is provisioned or at any time thereafter). A U-Prove information card can support any number of claims of any type. A claim value is presented to a Relying Party by presenting a UPT encoding the claim."(Paquin, 2010)
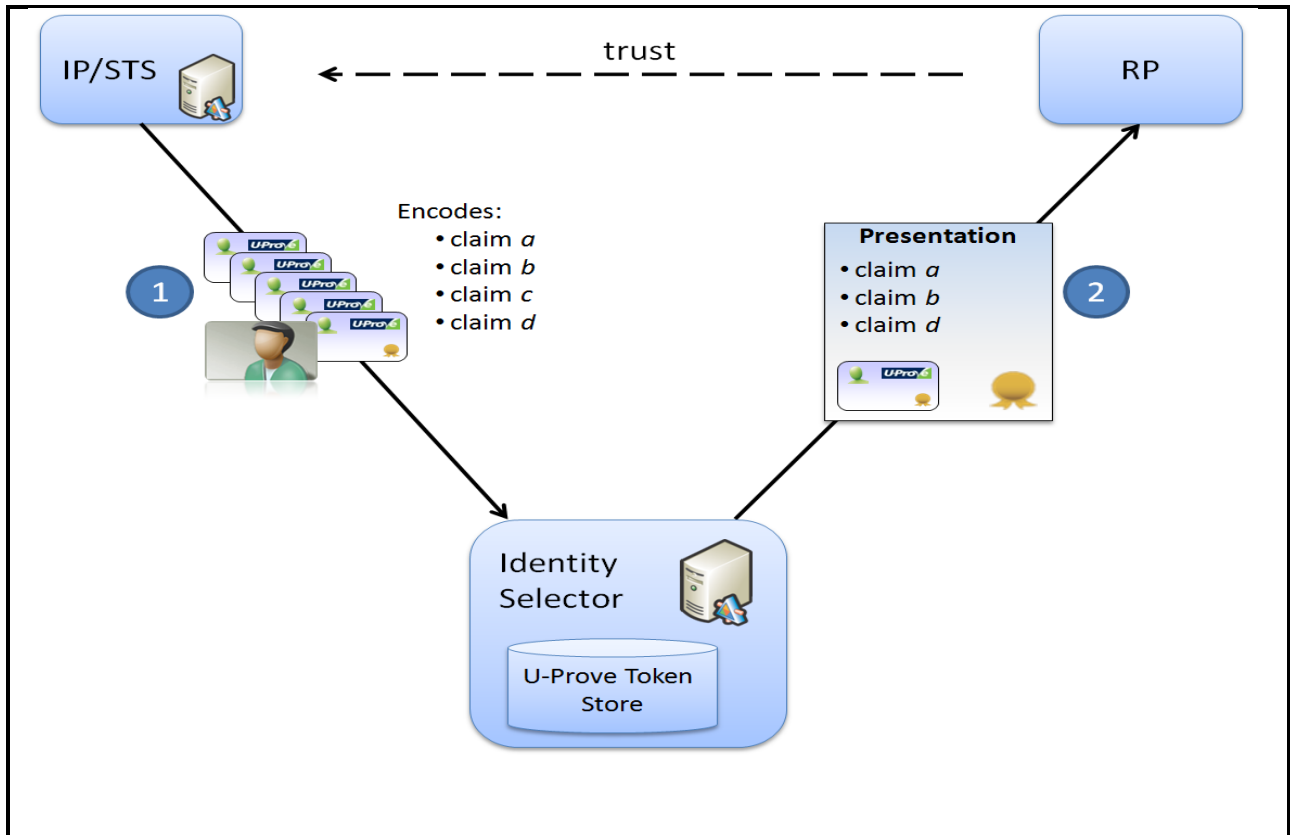
Figure 7 U-Prove Information Card (Paquin, 2010)

Chapter 3 – Methodology

The methodology adapted for this paper is Design Science Research. According to Hevner, March, Park, and Ram (2004) "the result of result of design-science research in IS is, by definition, a purposeful IT artifact created to address an important organizational problem."

This goal of this project is to research the Windows Cardspace development environment within the Microsoft .NET platform. In line with Design Science Research methodology the artifacts produced will include tutorials and practical labs which will serve to inform a programmer who has no experience of the subject matter.

Research Location

The research for this project was conducted primarily through the use of on-line sources. The major sources utilized were Books 24x7, IEEE Explore and ACM Digital Library. The Microsoft website has also provided numerous white papers and code sample kits for development of prototypes.

Instruments and Materials

The following instruments and materials were utilized in this project:

- All examples were developed under the Windows 7 Operating System
- Microsoft Visual Studio 2010 was utilized to develop all web site and service applications
- IIS was used as a hosting service on the local machine
- All presentations were developed using Microsoft PowerPoint 2003
- Code samples and SDKs downloaded from Microsoft website

Procedure

The procedure implemented to complete this project followed a process whereby research into Cardspace was conducted through the sources outlined above. The objective of the study is to develop labs and tutorials on the topic of Cardspace. Upon approval of the project statement research was conducted into the topic and sample applications were developed.

Visual Studio 2010 was installed and various aspects related to the development of the practical demonstration were developed, starting with developing knowledge of IIS, development of a web site with ASP.NET and development of a service example with Visual Studio.

Sample code including sample certificates were downloaded from the Microsoft website and installed locally. The requirement for sample code related to an application available in the development kit called tokenProcessor.cs. This code was required to decrypt the received security token. An in-depth description of this application is beyond the scope of this project – a high level description including a flow chart of the application is presented in the analysis section.

Evaluation

The artifacts developed by this project serve to introduce the Cardspace topic to the user. There are eleven tutorials developed including practical demonstrations. The breakdown of the Artifacts developed is as follows:

1. Introduction – introduces identity management and the requirements for an identity metasystem.

2. Creating an InfoCard – presents a practical example of developing an InfoCard

3. Cardspace – introduces the concept behind the InfoCard created in the previous step and also introduces the PKI (Public Key Infrastructure) system.

4. Identity Metasystem & Laws of Identity – introduces the reader to the concept of the identity metasystem and the Laws of Identity which form the basis for an identity metasystem.

5. Cardspace Model – presents the theory behind the Cardspace model in relation to the WS-* Specifications and how they are applied to Cardspace.

6. Configuring the Server & Installing Cardspace Sample Certificates – introduces the user to the server configuration for hosting the sample website and installation of the required certificates for the applications to be developed.

7. Triggering Cardspace – this tutorial presents an example of how the Cardspace interface is triggered by a website.

8. Unencrypted Token – this tutorial follows on from the Triggering Cardspace tutorial by presenting how to display the unencrypted token.

9. Decrypting a Token – this tutorial introduces the tokenProcessor software supplied by Microsoft , which detects the encryption used and then allows for decryption of the token and display of the required claims.

10. Setting up a Service with WCF and Cardspace – this tutorial presents a brief introduction to windows Communication Foundation (WCF) which allows for the development of a service through the WCF.

11. Cardspace and Services – the tutorial follows on from the previous tutorial and modifies the developed service to include Cardspace when requesting the required service.

12. U-Prove – this tutorial presents the topic of U-Prove which is now going to be Microsoft's identity management offering after retiring the Cardspace solution.

13. Developing with U-Prove – this tutorial presents an example based on Microsoft's whitepaper introducing U-Prove, showing how an application can be developed with Visual Studio.

The tutorials take the user through the Cardspace environment and start by introducing the concept of an identity management system and the requirements for such a system. This has been borne out of the increased reliance on the Internet by both business and consumers. The Laws of Identity are presented which form the basis of the Identity management system .The subsequent tutorials develop on this theory and present practical examples which show how to develop a website and a WCF service which triggers the Cardspace Identity Selector.

The interested reader is taken through the background for the Identity Metasystem and Laws of Identity and practical examples for developing Cardspace enabled sites and services. There is also a presentation on the latest identity technology offering from Microsoft with references for further reading for those that are interested in developing their knowledge of this topic.

Chapter 4 –Project Analysis and Results

This chapter will present an analysis of the practical applications developed in order to demonstrate the topic of Cardspace. The areas discussed in this chapter will be the

- The installation and creation of a card in Cardspace.

- The requirement to install and configure the IIS, certificates.

- The development of a website through Visual Studio 2010 which will trigger the Cardspace identity selector.

- The development of a service and modification of this service to trigger the Cardspace Identity selector.

- The introduction of the U-Prove technology and the SDK for this topic.

Installing Cardspace

The first step in the process is to ensure that Cardspace is installed locally – in Windows 7 and Vista (in XP Cardspace can be downloaded and installed) Cardspace can be run from the User Accounts option under the Control Panel – Figure 8 shows the location of Cardspace in the control panel window. Clicking on the Cardspace application will launch the Cardspace Identity Selector – this is shown in Figure 9. It can also be seen from this Figure that when Cardspace is loaded a private desktop is started.

Figure 8 Cardspace Application in User Accounts Setting Under Control Panel

There are a number of operations available from the Cardspace UI as presented in Figure

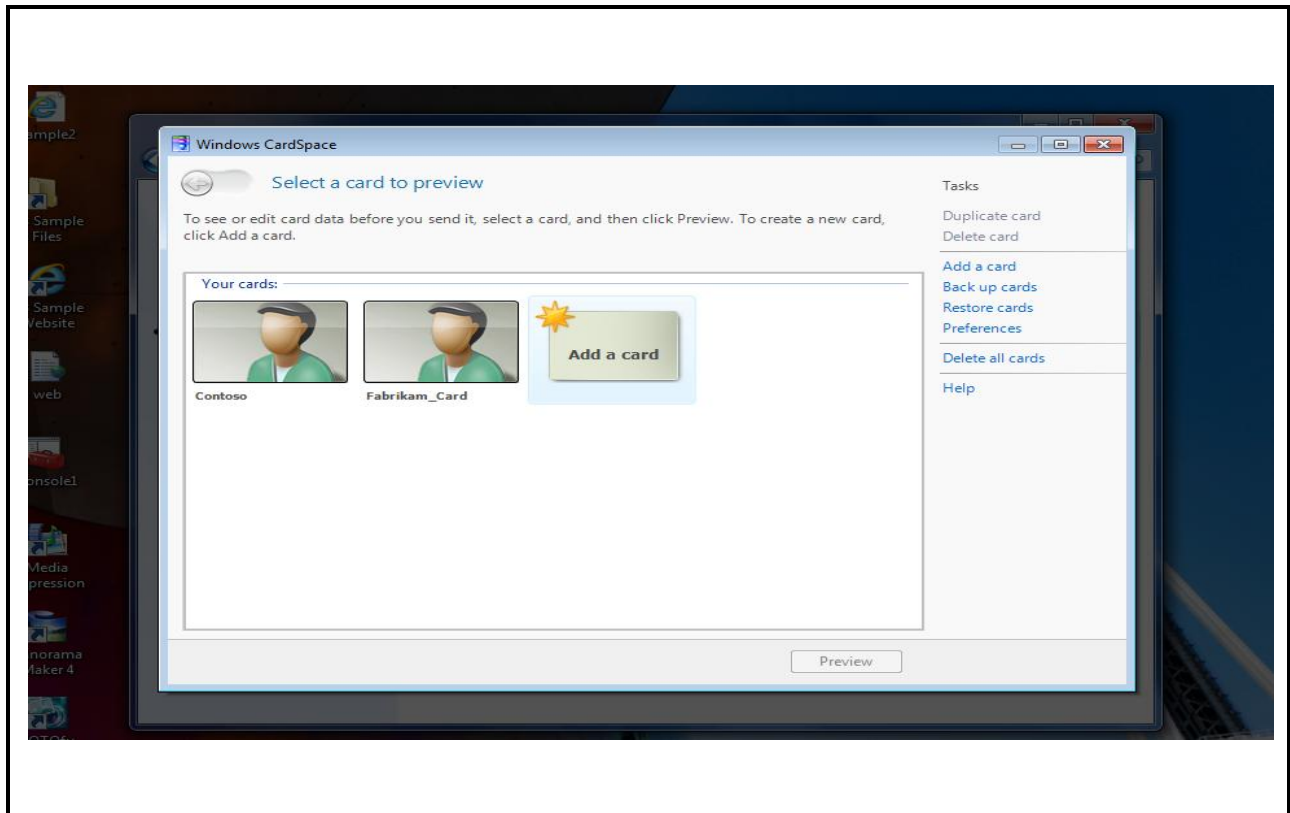9 – in the main section of the UI the list of cards available for the user will be displayed.

Figure 9 Cardspace Identity Selector Interface

The UI also presents the options such as the ability to add a card and back up cards. If Add a Card is selected the user is presented with two options - Create a Personal Card (Self Issued Card) or Install a Managed Card. The "cards" that are referred to here are also known as Info Cards. There are two types of cards that can be created – what is the difference?

"Every information card is created by some identity provider. For the self-issued identity provider, Cardspace provides a graphical tool that lets users create cards. For other identity providers, which typically will run on other machines, a user must acquire the appropriate cards in some way, such as through the identity provider's website, or through an e-mail message sent by the identity provider. How this is done is defined by each identity provider—there's no mandated way to acquire information cards." (Chappell, 2006)

"However it's acquired, each card (even one created by the self-issued identity provider) is digitally signed by the identity provider that issued it, and it comes with the identity provider's certificate. This signature is used to verify the identity of the identity provider itself. Once the card is on the user's machine, double-clicking it brings up a screen that allows the user to install this card into the standard Cardspace store. This is also when the user must approve the identity provider as a source for security tokens, as described earlier (although this approval isn't required for information cards created by the self-issued identity provider). Once the user has done this, the card can be used to request security tokens." (Chappell, 2006)

This section has presented the Cardspace Identity Selector and how this software is run showing that multiple cards can be stored in the Cardspace wallet. It also has displayed how the desktop is disabled when Cardspace is loaded – see Figure 9, this is performed in order to isolate the desktop. According to Bertocci et al., 2007 "every time Cardspace opens, all the applications already open on the user's desktop fade and cannot be accessed while Cardspace is up. It appears

that the applications have frozen. If you have the clock open on the taskbar, it will stay at the

time when Cardspace appeared. Was there a rift in time? What you are actually seeing behind the

Cardspace window is just a bitmap, a screenshot that was taken of the user's desktop. The bitmap

has been set in the background of a private desktop. Windows desktops provide isolation from

code running on other desktops. The most commonly seen desktop switch is from the default

desktop, where user applications run, to the win logon desktop. This switch is triggered by

pressing Ctrl+Alt+Del to lock your computer or enter your password."

Personal Cards (Self Issued Cards)

Personal Cards are cards that are created by the user locally using the Cardspace UI itself

– when selecting Create Personal Card the following screen, Figure 10 is displayed.
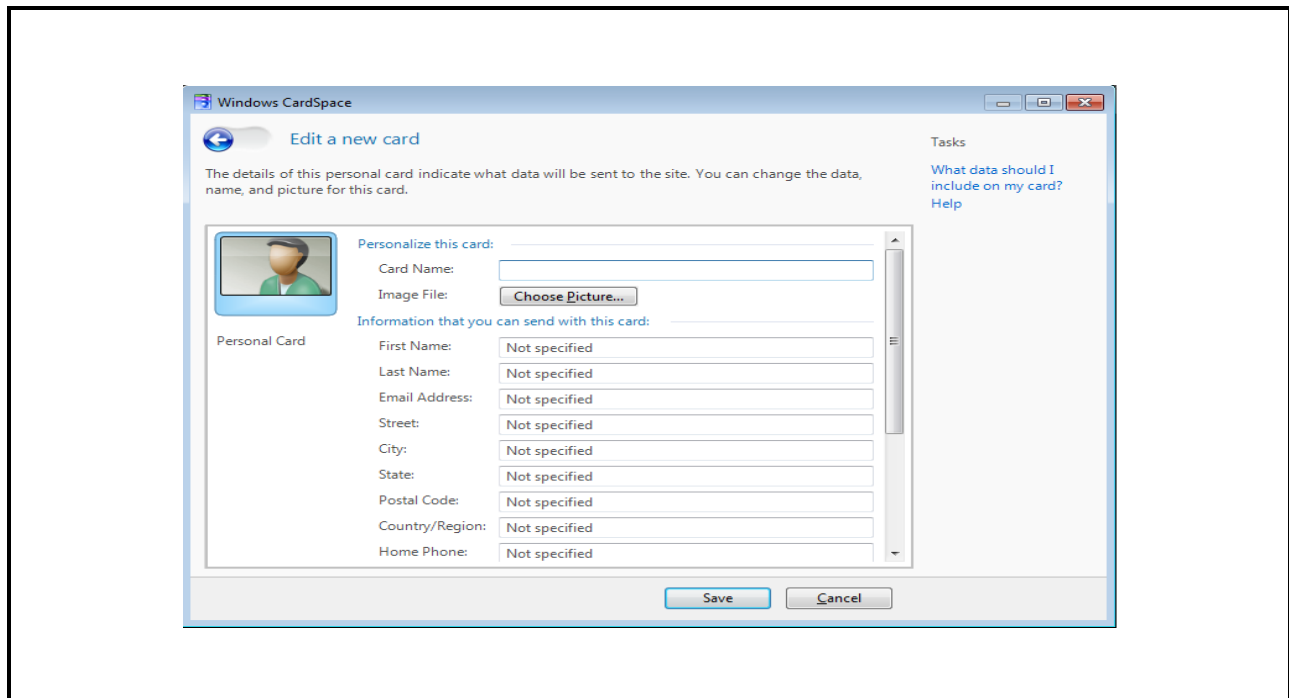
Figure 10 Creating a Personal Card

As can be seen from Figure 10, personal details can be entered for this card such as First name, Last Name, Email Address etc. When these details are entered the card is stored locally and available to be used for more than one website. When a website is accessed which requires an information card for logon, the Cardspace Identity Selector will be triggered and the cards that match the claims required by the website will be displayed. If there isn't a card in the wallet then the modifications (extra details) that are required by the website for a successful login will be highlighted in red in the Identity Selector.

Managed Cards

"Managed cards are a much more secure and controllable method of using information cards, because the cards are created and managed by a central, trusted, issuing party who can verify an individual's identity, rather than created by an individual. In addition, the issuer can associate any claim with the card, for example, employee number, department, security clearance level, etc. making the card more relevant to its purpose." (Avoco Secure, 2009)

"The claims associated with an issued card are managed and owned by the 'authority' which is the company that supplies the cards. The claims are managed and controlled by this authority and as such, can be changed and revoked as required, providing, potentially a mechanism for changing and revoking policies associated with resources such as websites and documents." (Avoco Secure, 2009)

The advantage of managed cards is that the amount of information transferred may be reduced eg. with managed cards there is a third party involved which the relying party trusts and if this relying party needs to know if the users credit rating is good or bad then this information may be transmitted from the third party. That is all that the relying arty needs to know – is the

credit rating good or bad. The third party or issuing party can determine if the users credit rating is good or not but the information transmitted to the relying party is minimized.

Information Card Contents

According to Chappell (2006), the contents of an information card help users intelligently choose a digital identity. They also allow Cardspace to match a card to a relying party's requirements, and to acquire an appropriate security token from the identity provider that issued this card. To accomplish these two goals, every information card contains the following:

- A JPEG or GIF file with the image of the card that the user sees on his or her screen, along with the name of the card that's displayed to him or her.

- One or more types of security tokens that can be requested from this identity provider, together with a list of claims each of those tokens can contain. This allows Cardspace to match a relying party's policy with the identity providers that can create security tokens meeting the relying party's requirements.

- A URL for one or more endpoints at this identity provider that can be accessed to request a security token.

- A URL identifying an endpoint at the identity provider from which its policy can be obtained. As described in the next section, this information also tells Cardspace how requests to the identity provider should be authenticated.

- The date and time the information card was created.

- A Cardspace reference for the card, which is a globally unique identifier specified as a URI. This identifier is created by the identity provider that issues the card, and it's passed back to that provider each time a security token is requested using this card.

"It's also important to note what's not in an information card: sensitive data about this identity associated with the card. For example, an information card created by a credit card company would not contain the user's credit card number. While this kind of sensitive information might appear as a claim in a security token created by an identity provider, it is always stored at the identity provider's system. When sent in a security token, this information is typically encrypted, making it inaccessible to both attackers and Cardspace. The key point is that sensitive information is never contained in an information card, and therefore it's never stored on the user's machine. If he or she chooses to, the owner of an information card can use Cardspace to preview the information that will appear in a security token created using that card. This information will be fetched from the identity provider that issued the card when the user asks to see it. Once it's been displayed, the sensitive information is then deleted from the user's system". (Chapell, 2006)

Cardspace Web Site Implementation

Cardspace may be triggered by a Website which is hosted locally through IIS. The web site may be developed through VS2010 – the first step is to create an empty website project under VS2010. The creation of this website will create a Default.htm file – in order for the website to trigger the Cardspace application, it is triggered through the use of an object tag in the webpage which supports an ActiveX application.

A Cardspace Object is created with the following three parameters:

tokenType – identifies the type of token – in the following example the token specified conforms to SAML 1.0

issuer – defines the issuer (identity provider) of the token – in this example the issue will be a self issued card

requiredClaims – the claims required by the website for authentication – the selected card to be sent will have to satisfy these claims. If a card does not exist then an existing card may be modified or a new card generated.

The code presented in Figure 11 is a sample of the code required to trigger the Cardspace identity selector. The code includes:

An object element which specifies the application type as an informationcard and assigns this instance of the application as xmlToken – this is specified on the line

<object type="application/x-informationcard" name="xmlToken">

The param name tokenType is specified as SAML 1.0 – this is specified on the line

<param name="tokenType" value="urn:oasis:names:tc:SAML:1.0:assertion" />

The param issuer is specified as self on the line

<param name="issuer"

value="http://schemas.xmlsoap.org/ws/2005/05/identity/issuer/self" />

The required claims by this website are specified in the RequiredClaims element and this website specifies that the claims givenmane, surname, locality and country must be presented in the info card. This specified the requirement that a card containing these claims must exist in the Identity Selector when the user navigates to the website.

The Default.htm webpage posts data to the aspx page placeholder.aspx – this is specified in the line

<form method=post action=placeholder.aspx>

The placeholder.aspx page is a simple page which will display a simple message

indicating that the Card has been submitted. Figure 12 presents the code for this page and shows

that just a simple message Card has been submitted is displayed.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>Active X triggered identity selector</title>
</head>
<body>
    <form method=post action=placeholder.aspx>
    <input  type="submit"  "Submit Information Card" />
        <object type="application/x-informationcard"
                name="xmlToken">
            <param name="tokenType"
            value="urn:oasis:names:tc:SAML:1.0:assertion" />
            <param name="issuer"
value="http://schemas.xmlsoap.org/ws/2005/05/identity/issuer/self
    " />
            <param name="requiredClaims"

    value="http://schemas.xmlsoap.org/ws/2005/05/identity/claim
    s/givenname

    http://schemas.xmlsoap.org/ws/2005/05/identity/claims/surna
    me

    http://schemas.xmlsoap.org/ws/2005/05/identity/claims/local
    ity

    http://schemas.xmlsoap.org/ws/2005/05/identity/claims/count
    ry"
            />
        </object>
    </form>
</body>
</html>
```

Figure 11 Default.htm code

```
<%@ Page Language="C#" AutoEventWireup="true"
    CodeFile="Placeholder.aspx.cs" Inherits="Placeholder"
    ValidateRequest="false" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
    <div>
        Card has been submitted
    </div>
    </form>
</body>
</html>
```

Figure 12 Placeholder.aspx code

The application is built under VS2010 and IIS is configured to run this application.

Opening IE7 and navigating to www.fabrikam.com will display the default webpage as shown in

Figure 13.

Clicking on the submit query button will trigger the Cardspace identity selector – if a

card is available which meets all the required claims, then that card will be available for

selection. If a card does not meet the requirements then this will be indicated on the desktop as

shown in Figure 14.

In order to satisfy the required claims the card may be edited to provide the requested

claims. The response will be submitted to the placeholder.aspx page where the html code present

in this file will display the message indicating that the card has been submitted to the user on

their browser.
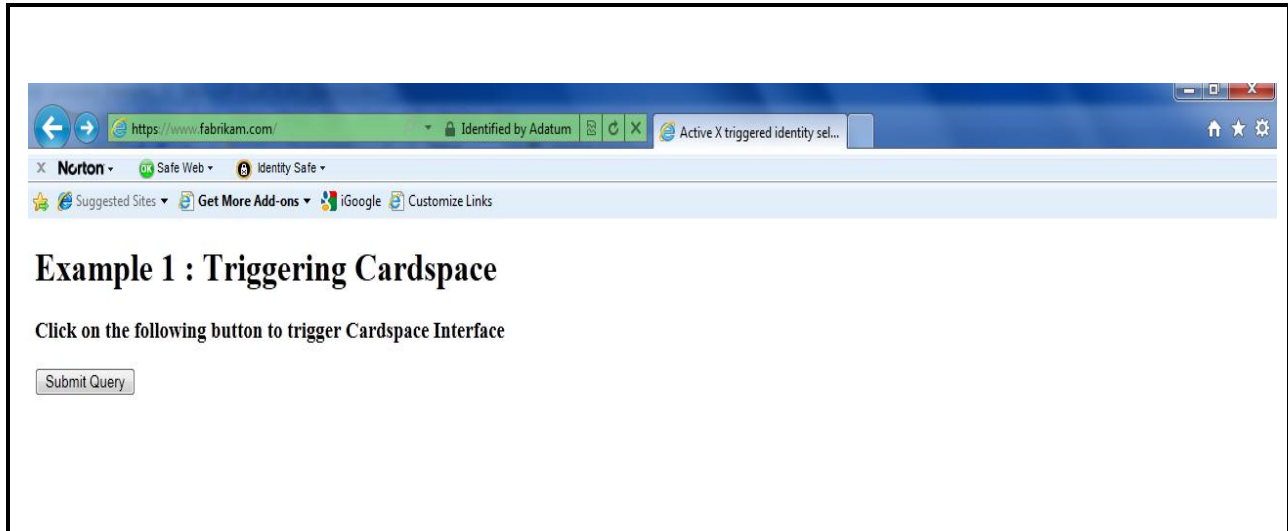
Figure 13 Default Webpage - Triggering Cardspace
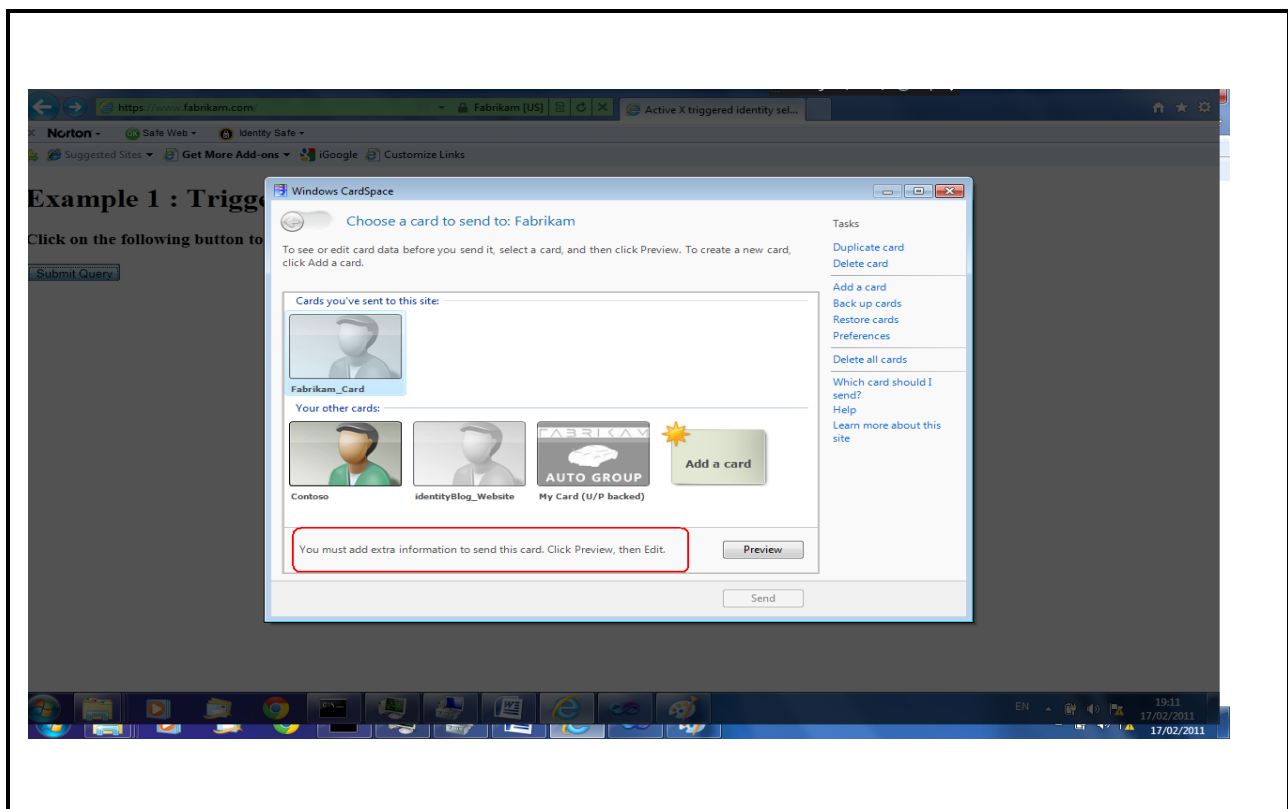
Figure 14 Cardspace indicating extra information is required

Displaying the Encrypted Token

The above example develops a website which triggers the identity selector and acknowledges the receipt of the information card. The next step in the process is to display the encrypted token and then to decrypt the claims from the token.

In order to display the token received, the application created in the preceding section will need to be modified. The first step is to add a new .aspx form to the application and to this form add a label and a textbox. The code for this form called DisplayEncryptedToken.aspx is presented in Figure 15. The default.htm page will need to be modified in order to call this page and the following code is to be added to the form load event for the DisplayEncryptedToken.aspx form.

```
public partial class DisplayEncryptedToken : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        string xmlToken;
        xmlToken = Request.Params["xmltoken"];

        if (xmlToken == null || xmlToken == "")
        {
            xmlToken = "N/A. No token was provided.";
        }

        lblEncryptedToken.Text = xmlToken;
        txtTokenMessage.Text = xmlToken;

    }
}
```

Figure 15 Code for DisplayEncryptedToken.aspx event

The xml token is defined as a string and retrieved from the Request.Parms collection. If there is no data stored in the xmlToken then a message indicating no token was provided is displayed otherwise the xmltoken is stored into the variables for storing data to the label and testbox created in the form displayEncryptedtoken.aspx as can be seen from the code presented in Figure 16. When this code is run then the following screen is displayed:
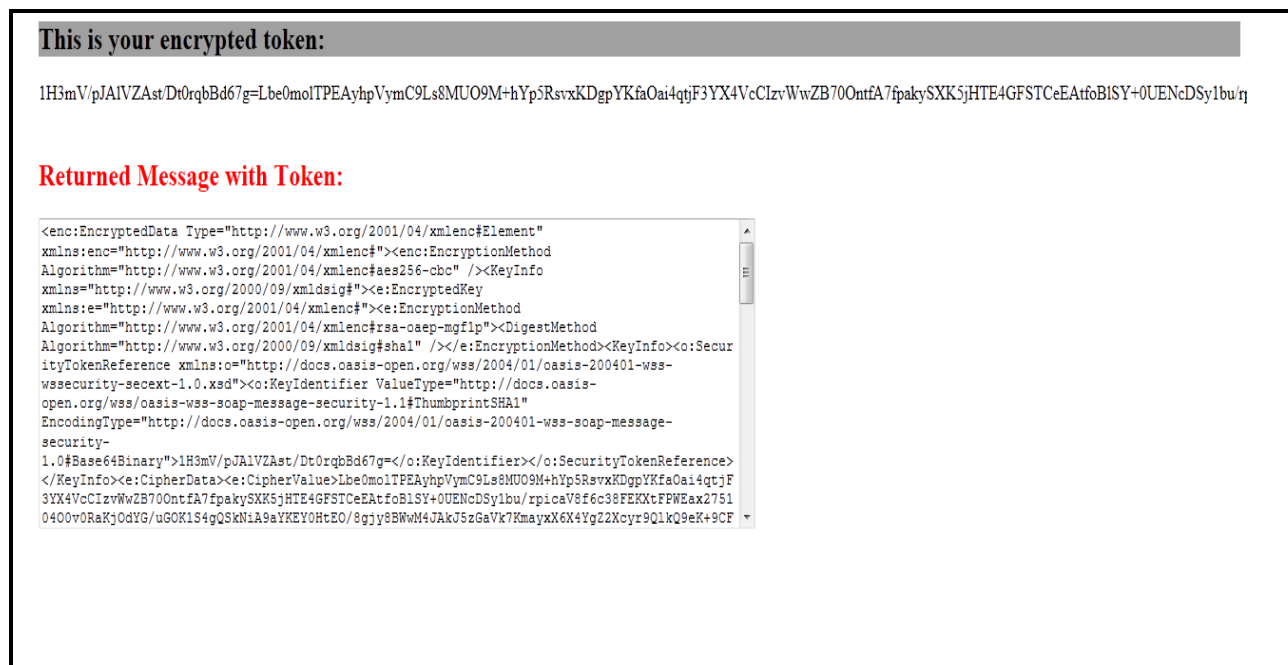
Figure 16 Displayed Encrypted Token.

Analysis of Returned Message

The section will analyze the encrypted token as displayed in Figure 16 and describe the elements contained within the returned message. The message is retrieved from the developed application and the analysis of the message is performed in conjunction with the article Decrypting a Security Token from Microsoft (Microsoft, 2007).

<enc:EncryptedData Type="**http://www.w3.org/2001/04/xmlenc#Element**"

xmlns:enc="**http://www.w3.org/2001/04/xmlenc#**">

```
      <enc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#aes256-cbc" />

        <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">

          <e:EncryptedKey xmlns:e="http://www.w3.org/2001/04/xmlenc#">

            <e:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p">

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

            </e:EncryptionMethod>

      <KeyInfo>

      <o:SecurityTokenReference xmlns:o="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-

      secext-1.0.xsd">

          <o:KeyIdentifier ValueType="http://docs.oasis-open.org/wss/oasis-wss-soap-message-security-

      1.1#ThumbprintSHA1" EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-

      message-security-1.0#Base64Binary">1H3mV/pJAlVZAst/Dt0rqbBd67g=</o:KeyIdentifier>

            </o:SecurityTokenReference>

      </KeyInfo>

<e:CipherData>


      <e:CipherValue>YGkuwQoBG4d0z3BZf+hDT398Ytf0yKVOz3KoIsEMCI87uCPh/zCEi3OM5yTYUf46nq8jA

      KGDPMTwgr/St8em9VmSc/AaahKYPxRE+BJV2bjT7wCRJgbmhBZN97RypdSzaMUIZ6qFt+pE+UOONpt

      3QRWIQo7reaOCs4PgOCKrUD5u7MPGhWLpEGkC8KP2TASHzgrHDkLSmJzSJKXmHY0H7BfWtiJFiv

      J+yL6NVIGjM8un5lPo5WVz0xUuFeA9N22mdG1P00DWSui0dOZyWmvYjFEi1oPPS9VkEJhNTxF7upZW

      zm1IV2iwEG7omZ1DFQK9EkWiP4RN1un6iBl+qD3bBw==</e:CipherValue>

      </e:CipherData>

      </e:EncryptedKey>

      </KeyInfo>

<enc:CipherData>


      <enc:CipherValue>blD7ZY8+Wbvw9KhrdiSWhB9BL4e1L+…………………………………………………………

      …………………………………………………………………………………………………………………………………………
```

…/N2ShrNCIiwMgYBNN+xqKlVIH/H/HG9n6zY07ED4bp6+PV7Om4BS/FhadGakUT21ryIBrCy2dJJ4Zoui

KsNJyvV+PzG2RPUG4ypL7NyLkDe

</enc:CipherValue>

</enc:CipherData>

</enc:EncryptedData>

The root element in the XML is <enc:EncryptedData> which contains the following

elements:

<enc:EncryptionMethod />

<KeyInfo />

<enc:CipherData />

The <enc:EncryptionMethod /> contains the algorithm used to encrypt the token in the

following line:

<enc:EncryptionMethod Algorithm="**http://www.w3.org/2001/04/xmlenc#aes256-cbc**" />

In this case the algorithm used is AES256-cbc – this is a symmetric key algorithm - in

which both parties use the same key for encryption and decryption. The symmetric key is

generated randomly, encrypted and stored in the <e:EncryptedKey> element which is contained

within the <KeyInfo /> element. The following is the data from the above message:

</KeyInfo>

<e:EncryptedKey>

<e:CipherData>

<e:CipherValue>**YGkuwQoBG4d0z3BZf+hDT398Ytf0yKVOz3KoIsEMCI87uCPh/zCEi3OM5yTYUf**

46nq8jAKGDPMTwgr/St8em9VmSc/AaahKYPxRE+BJV2bjT7wCRJgbmhBZN97RypdSzaMUIZ6q

Ft+pE+UOONpt3QRWIQo7reaOCs4PgOCKrUD5u7MPGhWLpEGkC8KP2TASHzgrHDkLSmJzS

JKXmHY0H7BfWtiJFivJ+yL6NVIGjM8un5lPo5WVz0xUuFeA9N22mdG1P00DWSui0dOZyWmvY

jFEi1oPPS9VkEJhNTxF7upZWzm1IV2iwEG7omZ1DFQK9EkWiP4RN1un6iBl+qD3bBw==

</e:CipherValue>

</e:CipherData>

</e:EncryptedKey>

</KeyInfo>

The encryption method used for encrypting the key is found in the following data

segment:

<e:EncryptionMethod Algorithm="**http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p**">

<DigestMethod Algorithm="**http://www.w3.org/2000/09/xmldsig#sha1**" />

</e:EncryptionMethod>

The encryption method algorithm used is rsa-oaep-mgf1p. The key for this operation is

found in the <KeyInfo /> element as follows:

<KeyInfo>

<o:SecurityTokenReference xmlns:o="**http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-**

wssecurity-secext-1.0.xsd">

<o:KeyIdentifier ValueType="**http://docs.oasis-open.org/wss/oasis-wss-soap-message-security-**

1.1#ThumbprintSHA1" EncodingType="**http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-**

soap-message-security-1.0#Base64Binary">**1H3mV/pJAlVZAst/Dt0rqbBd67g=**</o:KeyIdentifier>

</o:SecurityTokenReference>

</KeyInfo>

The <o:KeyIdentifier> allows the sender to inform the relying party which certificate's

key was used to encrypt the symmetric key, by putting its thumbprint (base64 encoded) into the

element. The relying party then uses its private key to decrypt the data in the <e:CipherData>

<e:CypherValue> element. Once the symmetric key is retrieved then the token is decrypted using the symmetric key.

Figure 17 presents the code generated to display the screen presented in Figure 16. The form displays the screen banner as Example 2: Encrypted Token under the h1 html element. Various other headings as displayed in Figure 16 are added under the html heading elements and the encrypted token is displayed in the asp:textbox.

Decrypting the Token

As stated above the symmetric key must be retrieved first before the token can be decrypted. Once this is performed then the token may be decrypted. The token is decrypted using an application called TokenProcessor.cs which has been downloaded from the Microsoft website. In order to decrypt the Token, components available in the .NET framework will be required. According to the .Net Framework Class Library (nd), the .NET framework descriptions provide the following components in order to decode the Token:

- System.ServiceModel - Contains the classes, enumerations, and interfaces necessary to build service and client applications that can be used to build widely distributed applications.

- System.IdentityModel - The System.IdentityModel namespaces contain types that are used to provide security and authentication in Windows Communication Foundation (WCF).

```
<%@ Page Language="C#" AutoEventWireup="true"
    CodeFile="DisplayEncryptedToken.aspx.cs"
    Inherits="DisplayEncryptedToken" ValidateRequest="false" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
        <div>

            <h1 style="color:Green">Example 2 : Encrypted Token</h1>
                <br />
                <br />
            <h2 style="background-color:ButtonShadow">This is your
    encrypted token:</h2>
        </div>

 <p style="height: 41px">
                <asp:Label ID="lblEncryptedToken" runat="server"
  Text="Label" Height="40px"
                Width="40px"></asp:Label>
        </p>
                <h2 style="color:Red"> Returned Message with Token:</h2>
            <p>
                <asp:TextBox ID="txtTokenMessage" runat="server"
  Height="240px"
                TextMode="MultiLine" Width="748px"></asp:TextBox>
            </p>
    </form>
</body>
</html>
```

Figure 17 DisplayEncryptedToken.aspx code

- System.IdentityModel.Selectors - Contains the classes that are used to provide security tokens for outgoing SOAP messages and authenticate security tokens in incoming SOAP messages. The SecurityTokenProvider class represents the base class for the classes that provide security tokens for an outgoing SOAP message. One of the classes that derive from the SecurityTokenProvider class is the X509SecurityTokenProvider, which provides an X509SecurityToken security token for outgoing SOAP messages. The SecurityTokenAuthenticator class represents the base class for the classes that authenticate security tokens in incoming SOAP messages. One of the classes that derive from the SecurityTokenAuthenticator class is the X509SecurityTokenAuthenticator class, which authenticates X509SecurityToken security tokens in incoming SOAP messages.

In the VS2010 application, references to the above components must be added and the TokenProcessor.cs namespace must also be imported. Figure 18 presents the .aspx code for diaplaying an unencrypted token. The import namespaces are specified at the top of the code and this code references a DisplayUnencryptedClaims.aspx.cs file which contains the code to extract the claims from the code returned from the call to the method in the TokenProcessor.cs code.

Figure 19 presents the code in this .cs file. This code creates a token object from the token class in TokenProcessor.cs. The token class in TokenProcessor.cs will validate the token and then decode the claims from the token.

. A flowchart analysis for this class is presented in Figure 20. The xmlToken is passed to this class which is first decrypted and stored into a byte array. This array is then deserialized and stored into m_token. An authenticator object is created and the token is validated using this authenticator object and m_token. If the token can be authenticated then the claimset is retrieved

```
<%@Import Namespace="System.IdentityModel" %>
<%@ Import Namespace="System.IdentityModel.Claims" %>
<%@ Import Namespace="Microsoft.IdentityModel.TokenProcessor" %>

<%@ Page Language="C#" AutoEventWireup="true" CodeFile="DisplayUnencryptedClaims.aspx.cs"
    Inherits="DisplayUnencryptedClaims" ValidateRequest="false" Debug="true" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
    transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body style="height: 218px">
    <form id="form1" runat="server">
    <div>
        <h1 style="color:Green">Example 3 : Decrypted Token</h1>
        <br />
        <h2 style="color:Red"> Token Claims:</h2>
        <br />
    </div>
    <asp:TextBox ID="tbClaims" runat="server" Height="285px" TextMode="Multiline"
        Width="737px"></asp:TextBox>
    </form>
</body>
```

Figure 18 DisplayUnencryptedClaims.aspx Code

from the token. The claimset requires a parameter to be specified in the application – this parameter, which is called IdentityClaimType, is specified in the web.config file in the application.

Figure 21 presents the appSettings section from the web.Config file and contains the IdentityClaimType parameter which specifies the givename claim for this parameter – this parameter is then used in the code to search the claimset for the claim givenname. If this is not found then the token is rejected.

```
public Token(String xmlToken)
{
    byte[] decryptedData = decryptToken(xmlToken);

    XmlReader reader = new XmlTextReader(new StreamReader(new
    MemoryStream(decryptedData), Encoding.UTF8));
    m_token =
    (SamlSecurityToken)WSSecurityTokenSerializer.DefaultInstance.ReadToken(reader
    , null);

    SamlSecurityTokenAuthenticator authenticator = new
    SamlSecurityTokenAuthenticator(new List<SecurityTokenAuthenticator>
    (new SecurityTokenAuthenticator[]
      { new RsaSecurityTokenAuthenticator(),
          new X509SecurityTokenAuthenticator() }), MaximumTokenSkew);


    if (authenticator.CanValidateToken(m_token))
      { ReadOnlyCollection<IAuthorizationPolicy> policies =
          authenticator.ValidateToken(m_token);
          m_authorizationContext =
          AuthorizationContext.CreateDefaultAuthorizationContext(policies);
          FindIdentityClaims();
    }
    else
    {
        throw new Exception("Unable to validate the token.");
    }
}
```

Figure 19 Token Class in TokenProcessor.cs

As can be seen from Figure 21 other parameters may be specified specified under the appSettings element – in the above example these are StoreName and StoreLocation. These parameters are used in the code by the findCertificate class – Figure 22 presents a snapshot of the relevant code which processes these parameters within this class.

The FindCertificate class will search the certificate store and return the certificate associated with the website. The claimset is returned to the .aspx webpage and the claims are retrieved from this value as shown in Figure 23.
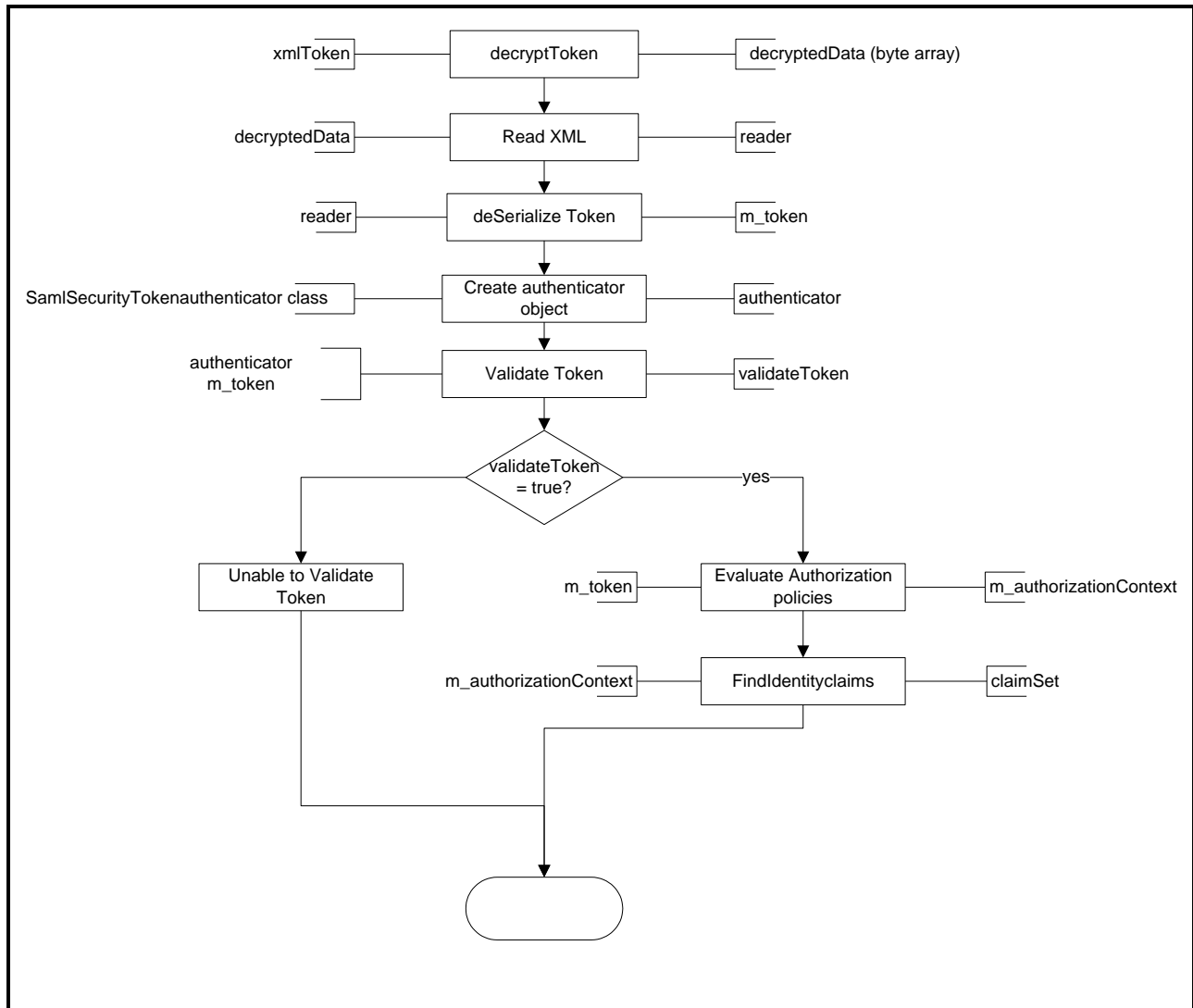
Figure 20 Token class flowchart

The code in figure 23 loops through the claims and builds a string which will be
displayed in the textbox on the website form. Figure 25 displays the claims returned. The claims
returned match the claims specified as required in the Default.htm webpage.

Figure 24 displays the claims specified in the Default.htm page – the claims required are
givenname, surname, locality and country. As shown in Figure 25 the values for each of these
required claims has been decrypted from the encrypted token.

```
<appSettings>
        <add key="StoreName" value="My"/>
        <add key="StoreLocation" value="LocalMachine"/>
        <add key ="IdentityClaimType"
        value="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/give
nname"/>
        <add key="MaximumClockSkew" value="60"/>
</appSettings>
```

Figure 21 Web.Config AppSettings

```
/// <param name="thumbprint">Certificate's thumbprint</param>
        /// <returns>the certificate, or the default certificate</returns>
        private static X509Certificate2 FindCertificate(object thumbprint)
        {
            X509Certificate2 result = m_certificates[thumbprint] as X509Certificate2;
            if (null == result)
            {
                StoreName storeName = StoreName.My;
                StoreLocation storeLocation = StoreLocation.LocalMachine;

                string rpCertificateThumbprint =
        System.Configuration.ConfigurationManager.AppSettings["CertificateThumbprint"];
                string rpStoreName = ConfigurationManager.AppSettings["StoreName"];
            string rpStoreLocation = ConfigurationManager.AppSettings["StoreLocation"];
```

Figure 22 findCertificate class code

```
{
    Token token = new Token(xmlToken);
    foreach (Claim claim in token.IdentityClaims)
    {
        tbClaims.Text += "Claim Type: " + claim.ClaimType + "\n";
        tbClaims.Text += "Right: " + claim.Right + "\n";
        tbClaims.Text += "Resource: " + claim.Resource.ToString() + "\n";
    }
}
}
}
```

Figure 23 Displaying Claims from returned claimset Code

```
<object type="application/x-informationcard" name="xmlToken">
    <param name="tokenType"
    value="urn:oasis:names:tc:SAML:1.0:assertion" />
    <param name="issuer"
    value="http://schemas.xmlsoap.org/ws/2005/05/identity/issuer/self" />
    <param name="requiredClaims"
    value="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/givenname
    http://schemas.xmlsoap.org/ws/2005/05/identity/claims/surname
    http://schemas.xmlsoap.org/ws/2005/05/identity/claims/locality
        http://schemas.xmlsoap.org/ws/2005/05/identity/claims/country"
    />
</object>
```

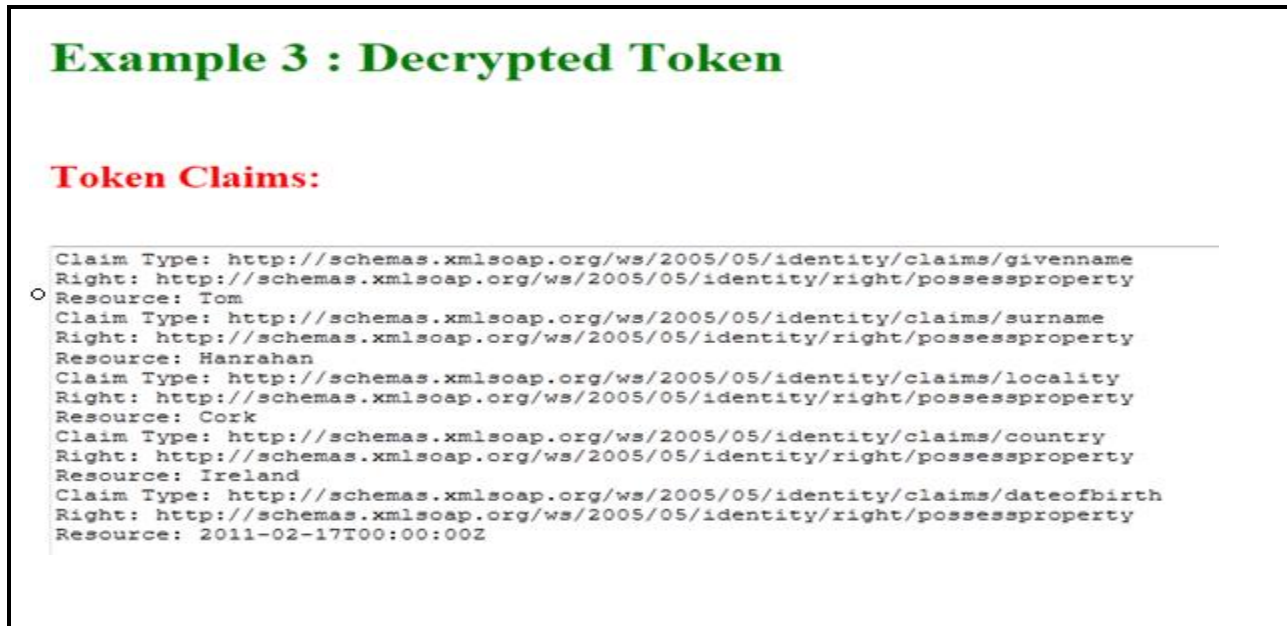Figure 24 Default.htm Specifying Required Claims

Figure 25 Decrypted Tokens

Cardspace Services Implementation

Cardspace can also be implemented in conjunction with services. WCF (Windows Communication Foundation) is a technology within the .NET framework which can be used to access services. Cardspace can be incorporated with these services to provide the identity requirements and developing Cardspace secured services.

WCF can be thought of in terms of ABC – Address, Binding and Contract

- Address is where the service is located eg. http://www.services.com/servicea

- Binding defines the how (address defines the where) – Binding contains information on the transport, security and encoding.

- Contract defies the operations of the service, the data that is exchanged by those operations and the fault messages returned when there is an error.

According to Mercuri (2007), advantages of the WCF are - the configuration is external to the code, ownership can be allocated where developers can concentrate on the operations and

messages while the IT department can be in charge/control over the address and binding

parameters. Should there be a need to change configuration values then this can be performed

externally to the code.

"From a hosting perspective, WCF provides tremendous flexibility. WCF services can

be hosted in any .NET application, which means anything from a console application to a

Windows Forms application to an NT service to a web server can host WCF".(Mercuri, 2007)

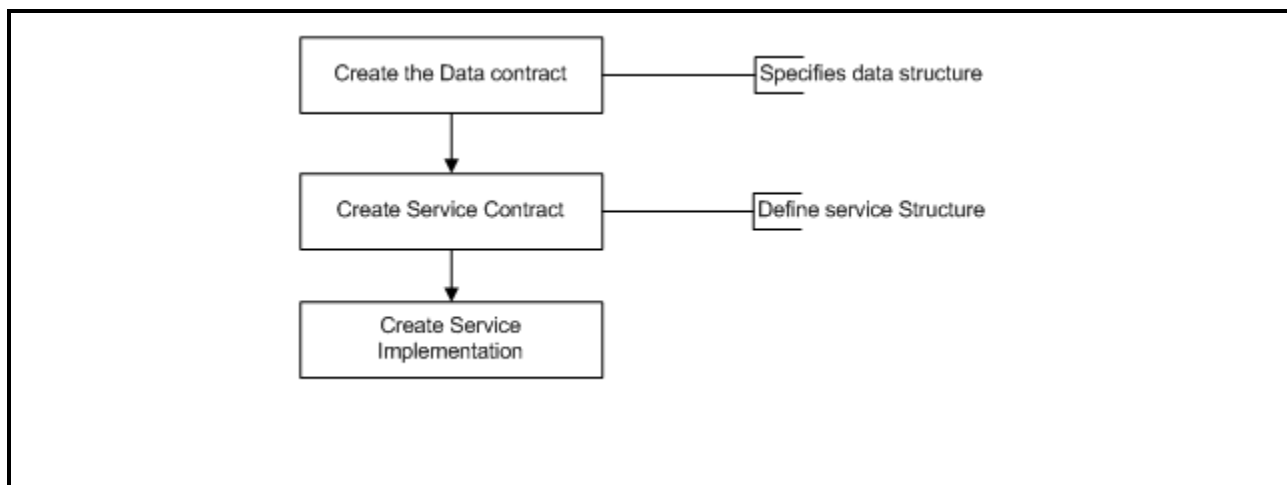Figure 26 presents an outline of the steps required in WCF to create a service.
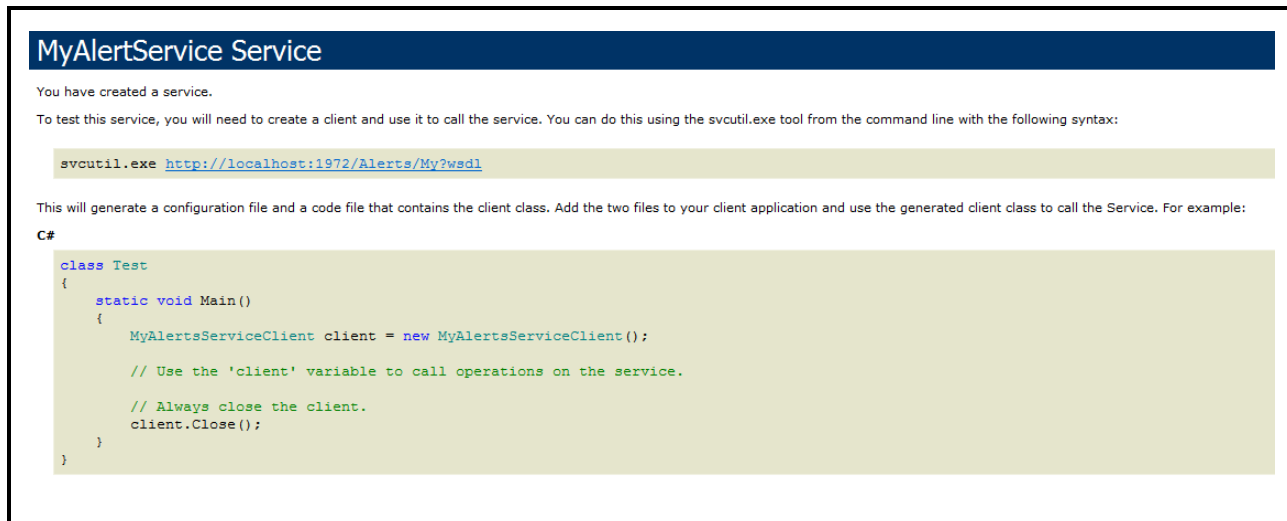
Figure 26 Service Creation Outline with WCF

The steps involved will result in a service implementation being created. The service

can be hosted in a number of ways, one method is to host the application as a console

application. In order to host a service, a service host has to be created and the bindings need to be

configured. Once the service has been hosted it can be tested which is performed by compiling

the application and then navigating to the service. Navigating to the service will display a

window which outlines a requirement to create the client. The client may be created with the

Service Metadata Utility (svcutil.exe).

The data presented by the Metadata screen as presented in Figure 27 outlines the steps required to generate the client by applying the svcutil and then suggesting the modifications required for the files in the application.

Figure 27 Metadata for Service

Cardspace and Services

The previous section outlined the requirements for generating a service. A service can be created which is secured with Cardspace. This involves creating a service which will reference the certificates required by the application and as a result the keys will be referenced which will allow for the decrypting of the required claims. The claims requested by a service will be configured in the app.config file by adding the required claims to a bindings element. Figure 28 presents the bindings element form the app.config file and shows that the addition of the claims requirements are specified in a similar fashion as in the web application.

```
<bindings>
    <wsFederationHttpBinding>
      <binding name="CardSpaceAlertsBinding">
         <security mode="Message">
           <message algorithmSuite="Basic128"
                      issuedTokenType= "urn:oasis:names:tc:SAML:1.0:assertion"
                      issuedKeyType="SymmetricKey">
              <issuer
  address="http://schemas.xmlsoap.org/ws/2005/05/identity/issuer/self" />
              <claimTypeRequirements>
                <add claimType=
  "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/givenname"/>
                <add claimType=
  "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/surname"/>
                <add claimType=
  "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/locality"/>
                <add claimType=
  "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/country"/>
              </claimTypeRequirements>
            </message>
          </security>
        </binding>
      </wsFederationHttpBinding>
 </bindings>
```

Figure 28 App.Config bindings element

Figure 28 presents the specification of the required claims – when the service is run the

information card is triggered by the presence of the service element in the app.config file. Figure

29 presents the code in this section.

```
  <service behaviorConfiguration="MyAlertsServiceBehavior"
 name="Services.MyAlertService">
         <endpoint address="MyAlerts" binding="wsFederationHttpBinding"
          bindingConfiguration="CardSpaceAlertsBinding"
    name="AlertsCardSpace"
          contract="ServiceContracts.IMyAlertsService">
           <identity>
             <certificateReference
                               storeName="My"
                               storeLocation="LocalMachine"
                               x509FindType="FindBySubjectName"
                               findValue="www.fabrikam.com"
               />
           </identity>
         </endpoint>
         <endpoint address="mex" binding="mexHttpBinding"
    contract="IMetadataExchange" />
       </service>
```

Figure 29 Service element in app.config for Service Implementation

U-Prove Implementation

In order to develop with U-Prove, Microsoft has provided a toolkit to allow developers

interact with this new technology. The basic requirements for this technology are:

- Microsoft Visual Studio 2010

- Windows Identity Foundation - http://msdn.microsoft.com/en-

  us/evalcenter/dd440951.aspx

- U-Prove CTP R2 WIF Extension

- U-Prove CTP R2 RP Toolkit

The **RP Toolkit** is a Visual Studio template to help with the development of U-Prove aware ASP.Net applications which requires Visual Studio 2010 Professional or higher. The **Windows Identity Foundation Extension** is an extension to the Windows Identity Foundation (WIF) that provides the ability to build a custom claims provider (a.k.a. Security Token Service or STS) for U-Prove token issuance and to enable relying parties to verify U-Prove token presentations.

The installation of the RP Toolkit and WIF Extension allow for the development of a U-Prove Relying Party Web Application with VS2010. This allows the creation of a basic website which will allow for the authentication against the hosted U-Prove agents and hosted Claims Providers.

"The Agent is composed of a cloud-hosted service and optional client components. The cloud-hosted Agent can be used with all major browsers on Windows, MacOS, and several smartphones. The first optional client component is a Silverlight component which enables local storage of U-Prove tokens and enhances the privacy and security for the user. The second optional component is an IE plugin that looks for a U-Prove Agent object tag in the RP page and manages the launch of the Agent to ensure the user's choice of agent, if one was made, is respected."(Brown et al., 2011)

In the generated U-Prove website, the URI for the U-Prove agent must be specified – this is specified in the UProveAgentSettings.cs. Edit this file and change the default value of Null to https://uprove-agent.cloudapp.net. The next step is to uncomment the RPPolicy from the RPPolicy.xml file under the solution explorer in VS2010.

This file specifies the required claims that the sample Relying Party has requested – this is presented in Figure 32. This is a similar to what has been presented already in Cardspace– this

```
<RPPolicy xmlns="http://schemas.xmlsoap.org/ws/2011/02/u-prove/policy">
  <Name>Sample Relying Party</Name>
  <RequestedClaims>
    <ClaimType Uri="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/givenname"/>
    <ClaimType Uri="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/surname"/>
    <ClaimType Uri="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/streetaddress"/>
    <ClaimType Uri="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/dateofbirth"/>
    <ClaimType Uri="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/postalcode"/>
  </RequestedClaims>
  <ClaimProviders>
    <ClaimProvider Uri="https://uprove-claimprovider.cloudapp.net"/>
    <ClaimProvider Uri="https://brown-cp.cloudapp.net"/>
    <ClaimProvider Uri="https://woodgrove-cp.cloudapp.net"/>
    <ClaimProvider Uri="https://green-cp.cloudapp.net"/>
    <ClaimProvider Uri="https://yellow-cp.cloudapp.net"/>
    <ClaimProvider Uri="https://purple-cp.cloudapp.net"/>
  </ClaimProviders>
</RPPolicy>
```

Figure 30 RPPolicy element in RPPolicy.xml

file specifies five required claim types. When this application is run, the screen presented

in Figure 31 is displayed. This screen displays the requested information that the Sample Relying
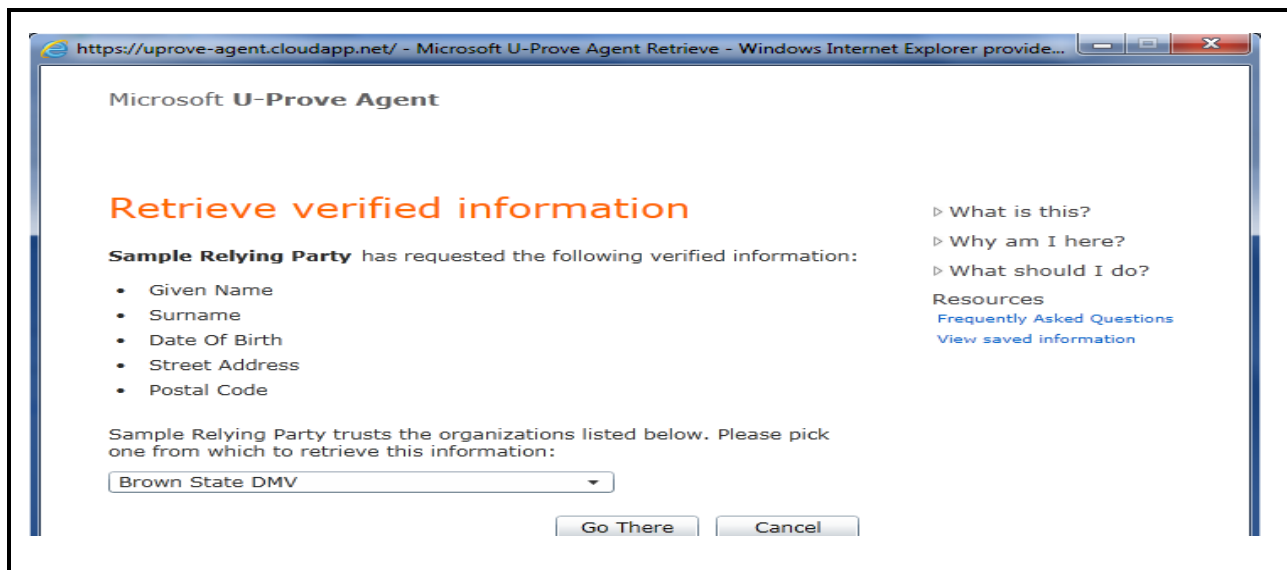

Figure 31 U-Prove Agent

Party has requested, which correlates with the information specified in the RPPolicy element as presented already. The Sample Relying Party also presents a list of the organizations which it trusts to retrieve the required information from. These claims providers are specified also in the RPPolicy.xml file under the ClaimProviders element – see Figure 30 for the ClaimProviders specified in this example.

Figure 32 Claims Provider

One of these organizations may be selected and the Claims Provider will be loaded as shown in Figure 32. This claims provider will require authentication for the user – this can be provided for the sample site by selecting one of the fictitious users on the right hand side of the screen. The claims provider will authenticate the user and return the required claims to the U-Prove agent. This screen is presented in Figure 33.
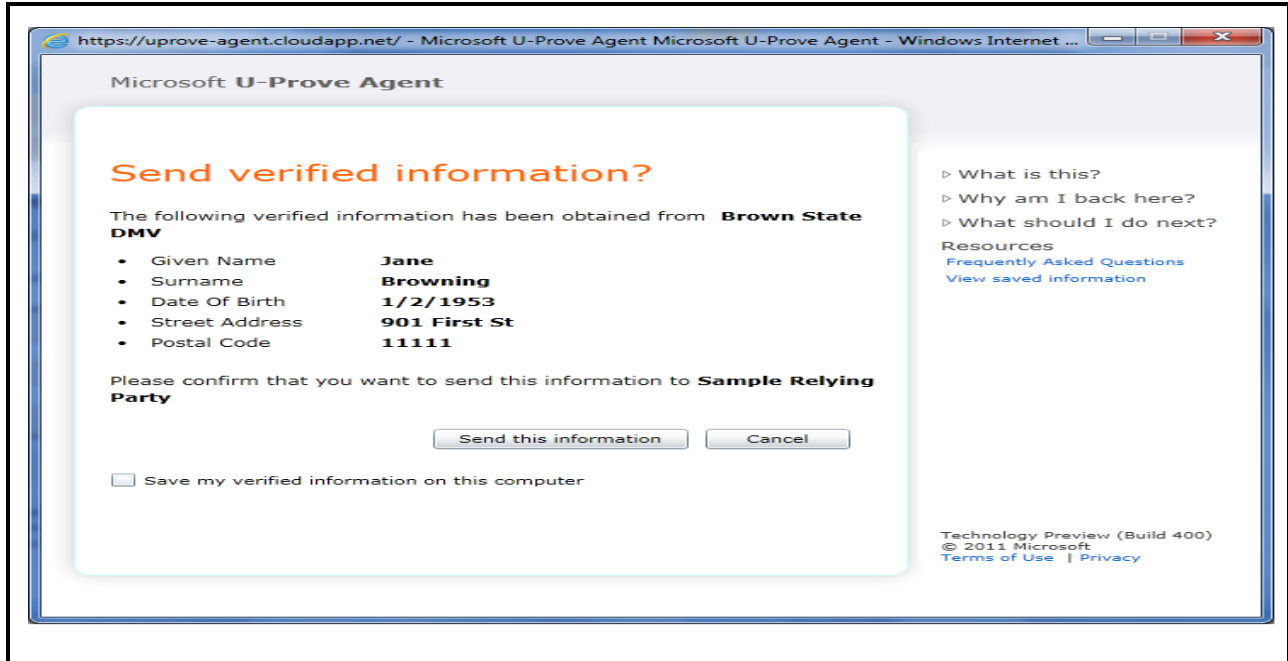
Figure 33 U-Prove Agent Request to Send

Figure 34 displays the claims requested for the selected user on the previous screen. The

user can select to send this information or Cancel. If the information is sent then the user will be

directed back to the RP and the claims will be displayed as shown in Figure 34.
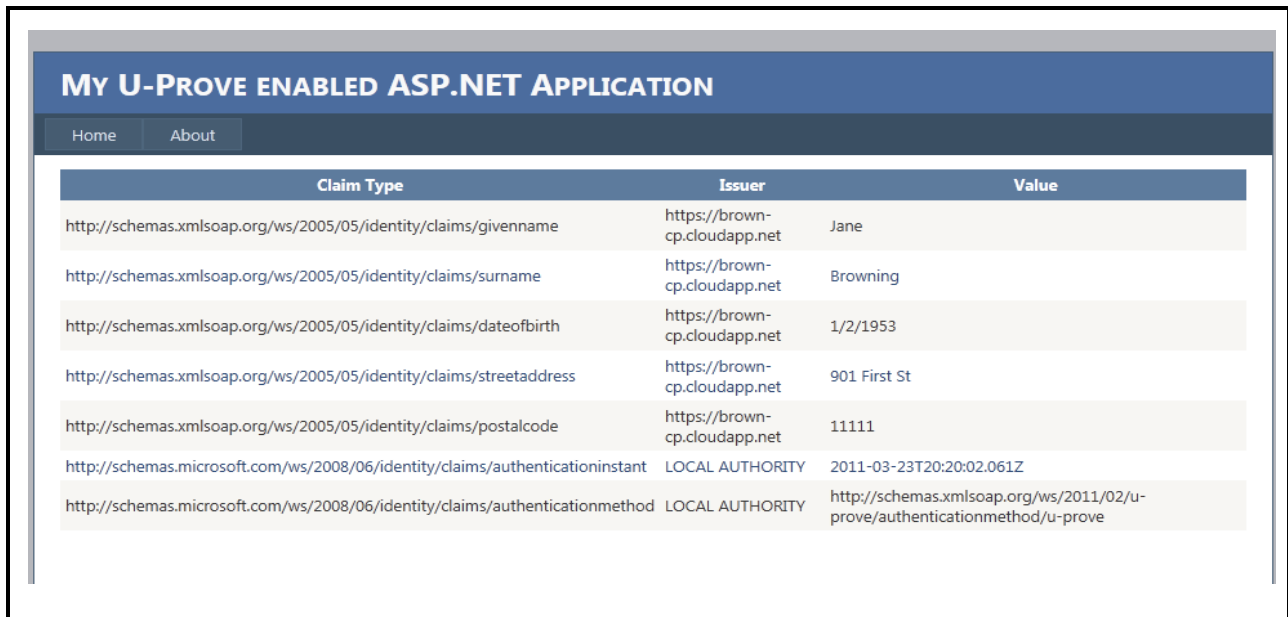
Figure 34 RP site displaying claims

The interactions between each of the interested parties in this application are presented in the swim lanes diagram in Figure 35.

Initially the user navigates to the RP page who is then redirected to the U-Prove Agent(UPA) which is specified in the RPPolicy element of the RPPolicy.xml file. The UPA gets the CP policy from the Claims Provider, who returns its CP policy.

The user is redirected to the CP authentication page which allows for the generation of the tokens by the CP. The tokens are communicated to the UPA by using WS-Trust. The tokens are then presented to the relying party.
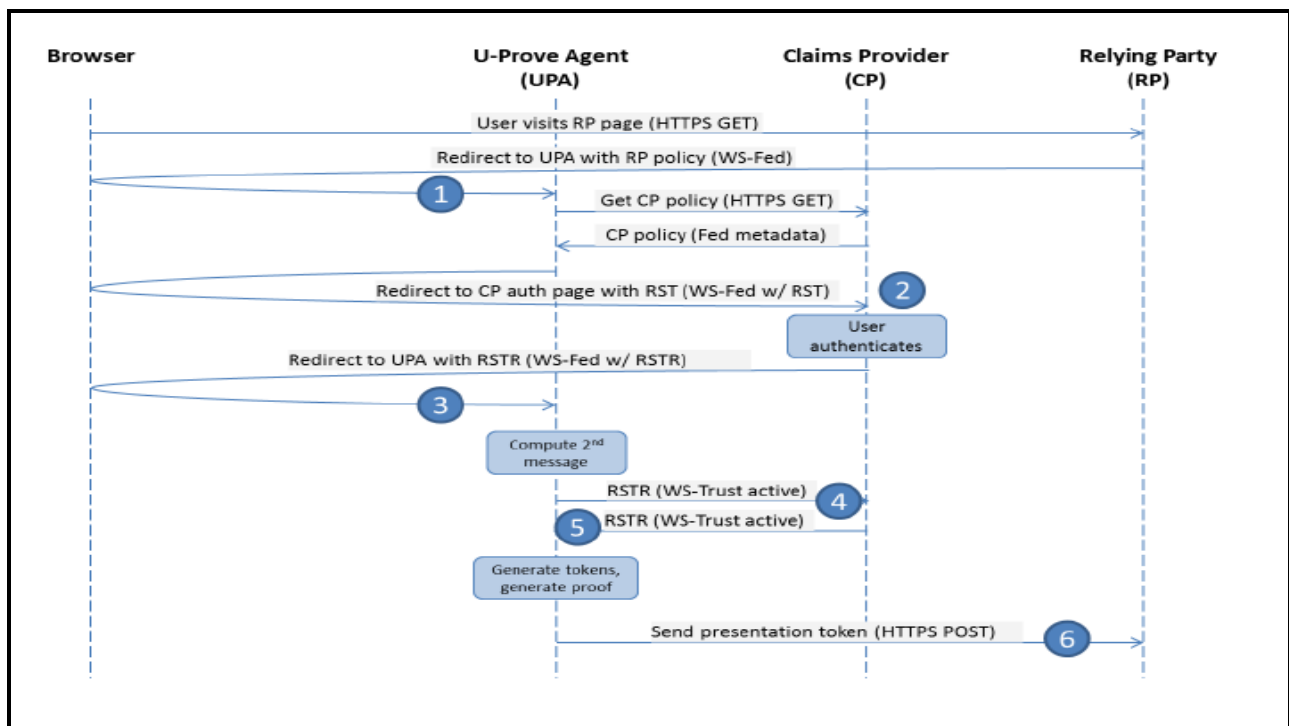
Figure 35 U-Prove Interactions (Brown et al., 2011).

U-Prove Architecture
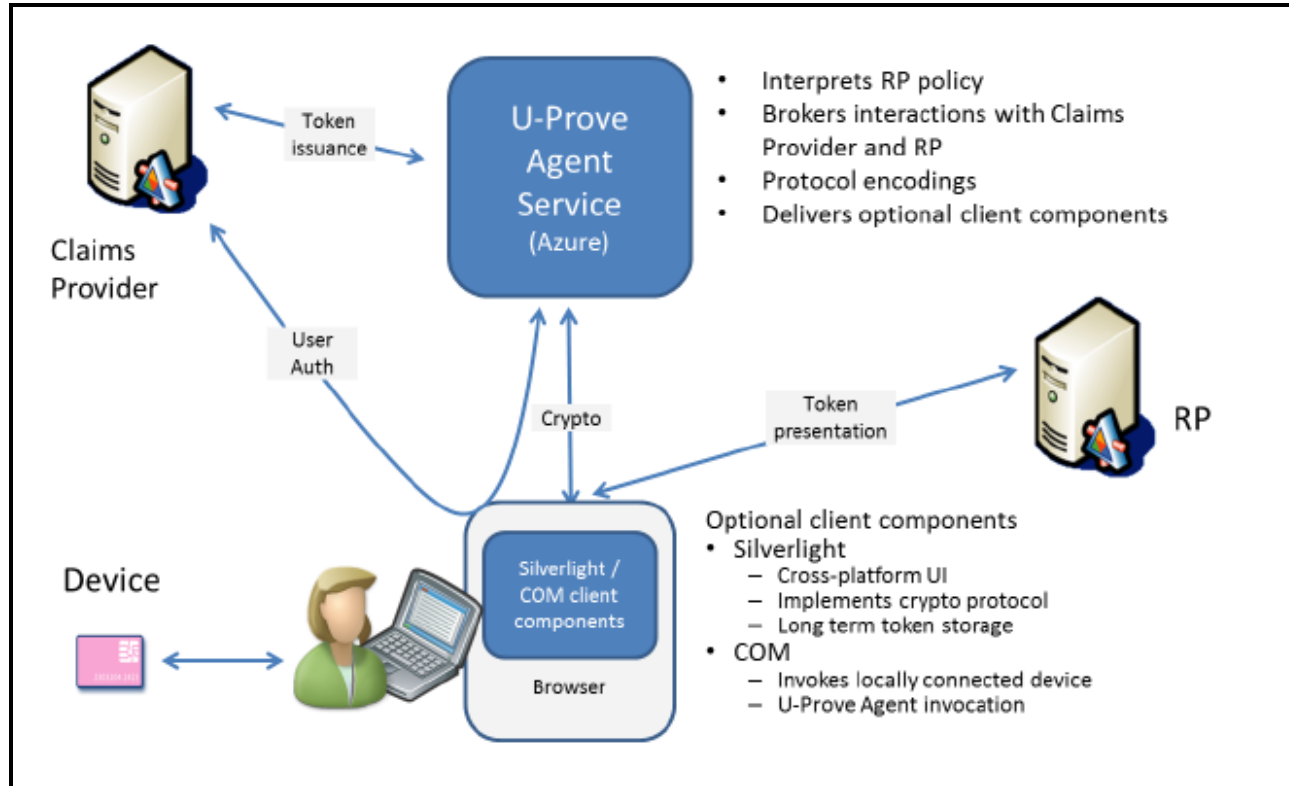
Figure 36 U-Prove Architecture (Brown et al., 2011).

Figure 36 presents the U-Prove architecture. "If the browser in use does not support Silverlight, or the user opts not to install it, the user will interact with an HTML-based cloud service. However, if Silverlight is installed the U-Prove cryptography will be performed on the user's machine, thus improving the overall security and privacy aspects of the solution. For device binding, the Silverlight component is required, as well as a COM component to interact with the hardware device.

The U-Prove Agent uses WS-Federation as a passive protocol to drive the initial interaction between the Relying Party and the U-Prove Agent and the user authentication between the U-Prove Agent and the Claims Provider. For the generation of the U-Prove Tokens

the calls between the U-Prove Agent and the Claims Provider use WS-Trust." (Brown et al.,

2011)

Conclusion

This chapter set out with presenting an analysis of the following objectives:

- The installation and creation of a card in Cardspace.

- The requirement to install and configure the IIS, certificates.

- The development of a website through Visual Studio 2010 which will trigger the

  Cardspace identity selector.

- The development of a service and modification of this service to trigger the

  Cardspace Identity selector.

- The introduction of the U-Prove technology and the SDK for this topic.

This chapter has built on the research material presented in Chapter 2 which presented the

theoretical background to Cardspace through the introduction of the Laws of Identity, the

Identity Metasystem and the application of the WS-* specifications. The analysis presented in

this chapter outlines the how to install the Cardspace and presented the topic of managed or self-

issued cards. The analysis continues with the development of a website showing how to trigger

the Cardspace Identity Selector and follows on to display the unencrypted token and then how to

retrieve the required claims from the encrypted token. The token is decrypted through the use of

the tokenProcessor code supplied by Microsoft.

The analysis of the website implementation outlines practical examples of the theory

presented in Chapter 2. Examples of the XML messaging have been presented with descriptions

of the XML encryption elements within these messages. It has also been presented how the encryption key information has been transmitted to the decryption process.

This chapter has shown how to install and trigger the Cardspace Identity Selector through a website or through a service developed with WCF. The chapter has also presented an overview of Microsoft's new Identity Management solution as Cardspace has been retired in 2011. Visual Studio is a requirement for the development environment for U-Prove and U-Prove leverages the cloud technology for the agents that form the core of this new technology. The major difference between the applications is the cryptographic technology developed by Credentica. Another difference is the absence of the Cardspace Identity Selector, with U-Prove having cloud based U-Prove agents.

This chapter has shown how to develop a website or service which can trigger the Cardspace Identity Selector and presented an analysis of the messages between the relevant parties. The next generation of Identity Management system from Microsoft based on the U-Prove technology has been introduced with an example from the current SDK. References have been included for further reading and application development for this technology.

Chapter 5 – Conclusions

The increasing use of applications and services on the Internet has resulted in the requirement for a federated identity system allowing users to share a single identity across multiple sites on the Internet. The increasing use of applications and services on the Internet requiring logon details has resulted in password fatigue due to the different logon requirements for each of these applications and services. The increasing problem of Identity theft has resulted in the introduction of different Identity management systems. There have been numerous attempts at resolving the identity issues with such systems as Open ID, Kerberos and Microsoft Passport. There has been much discussion in relation to this issue with Kim Cameron being a major driver in the development of the Laws of Identity and the Identity Metasystem. Microsoft's identity management solution was Cardspace which complies the Laws of Identity and the Identity Metasystem.

This project has researched Cardspace and presented an analysis of the practical implementations on Cardspace where Cardspace is a token based authentication system based on open XML standards. The implementation of Cardspace with Self-Issued and Managed cards through the Cardspace Identity Selector has been presented. The development of a website which triggers Cardspace identity selector has been presented and also the application of Cardspace to a WCF service application has been demonstrated. The detail behind the Cardspace Implementation has been considered with an introduction to such areas as Web Services, PKI and WS-*.

In February 2011, Microsoft announced that they would not ship Cardspace 2.0. Microsoft claim that Windows Cardspace was initially released and developed before the pervasive use of online identities across multiple services and the identity landscape has changed

with the evolution of tools and cloud services. Microsoft claim that claims-based identity

remains a central concept for Microsoft's identity strategy with the release of U-Prove. The

release of U-Prove will take the form of a user agent that takes account of cloud computing

realities and takes advantage of the high-end security and privacy capabilities within the

extended U-Prove cryptographic technology.

There is still a requirement for an Identity management system on the Internet as

password fatigue is still an issue. It is very difficult to manage all the different passwords for all

the different services – it may be that people use the same password across different sites or use

obvious passwords to help them remember the passwords. Either of these approaches are not

very secure. Cardspace was not successful and has not been taken up by the wider development

community – what system out there will be – will it be U-Prove? This project has considered

Cardspace and the principles that it has been founded upon and introduced the topic of Identity

management systems. It is hoped that the new technology will be successful and will be taken up

by the wider development community in order to offer enhance protection to the user of services

on the Internet especially with the development of the cloud computing era.

References

.Net Framework Class Library http://msdn.microsoft.com/en-us/library/gg145045.aspx

Avoco Secure, 2009 Windows Cardspace Information Cards and secure2trust retrieved

from http://www.avocosecure.com/media/htmlPages/products/Cardspace/ UsingWindows

Cardspaceandsecure2trust.pdf

Bertocci, V., Serack, G., Baker, C. (2007) Understanding Windows Cardspace An Introduction

to the Concepts and Challenges of Digital Identities Boston, MA : Pearson Education

Bocij, P. (2006) The Dark Side of the Internet : Protecting yourself and your family from online

criminals retrieved from http://books.google.ie/books?id=o5h2qwyDzRsC&dq

=identity+theft+on+the+Internet&source=gbs_navlinks_s

Britton, C., Bye, P. (2004) IT Architectures and Middleware Second Edition, Pearson Education:

Boston

Brown, J., Stradling, P., Wittenberg, C.H. (2011) U-Prove CTP R2 Whitepaper retrieved from

http://download.microsoft.com/download/0/9/8/0981B3E0-FDB4-4979-B799-

009C6885BB6F/U-Prove_CTP_R2_Whitepaper.pdf

Cameron, K. (2005) The Laws of Identity retrieved from

http://www.identityblog.com/stories/2005/05/13/TheLawsOfIdentity.pdf

Chappell, D. (2006) Introducing Windows Cardspace retrieved from

http://msdn.microsoft.com/en-us/library/aa480189.aspx

Charney S., (2009) The Evolution of Online Identity retrieved from

http://www.ieeexplore.ieee.org.libgate.library.nuigalway.ie/stamp/stamp.jsp?tp=&arnum

ber=5280133

Lakshminarayanan, S. (2008). Oracle web services manager: securing your web services

retrieved from http://common.books24x7.com.dml.regis.edu/toc.aspx?bookid=30175.

Leedy, P. D., & Ormrod, J. E. (2005). Practical Research: Planning and Design (8th ed.). Upper

Saddle River, NJ: Pearson Education.

Liberty, J., Horovitz, A. (2008) programming .NET 3.5 retrieved from

http://books.google.ie/books?id=YnPDNMy6-OkC&dq=Cardspace+WS-

Trust&source=gbs_navlinks_s

Hevner, A.R., March, S.T., Park, J., Ram, S. (2004) DESIGN SCIENCE IN INFORMATION

SYSTEMS RESEARCH MIS Quarterly Vol. 28 No. 1, pp. 75-105

Kantara Initiative retrieved June 22, 2011 from http://kantarainitiative.org/

Open ID retrieved June 22, 2011 from http://openid.net/

Madsen, P. (2004) Federated Identity and Web Services retrieved from

http://www.sciencedirect.com.dml.regis.edu/science/article/pii/S1363412704000329

Microsoft (2007) Decrypting a Security Token retrieved from http://msdn.microsoft.com/en-

us/library/aa967562(VS.90).aspx

Mercuri, M. (2007) Beginning Information Cards and Cardspace: From Novice to Professional

retrieved from http://library.books24x7.com.dml.regis.edu/assetviewer.aspx?

bookid=25510&chunkid=334407539

Paquin, C. (2010) U-Prove Technology Integration into the Identity Metasystem V1.0 retrieved

from https://connect.microsoft.com/site642/Downloads/DownloadDetails.aspx?

DownloadID=26953

Ryman, A. (2003) Understanding Web Services retrieved from

http://www.ibm.com /developerworks/websphere/library/techarticles/0307_ryman/ryman.html

U-Prove Community Technology Preview R2 (2011) retrieved from

http://connect.microsoft.com/site1188

Vedamuthu, A.S., Roth, D. (2006) Understanding Web Services Policy retrieved from

http://msdn.microsoft.com/en-us/library/ms996497.aspx

Web Services Architecture (2004) retrieved from http://www.w3.org/TR/ws-arch/#gengag

XML-Signature Requirements, W3C Working Draft 14-October-1999 retrieved from

http://www.w3.org/TR/xmldsig-requirements#Model

Annotated Bibliography

Maler, E., Reed, D. (2008) The Venn of Identity Options and Issues in Federated Identity Management. Retrieved from IEEE Security and Privacy, March 2008 pp 16-23

This paper introduces federated identity management. **Federated identity management** is a set of technologies and processes that let computer systems dynamically distribute identity information and delegate identity tasks across security domains. Web applications can offer users **cross-domain single sign-on** (SSO), which lets them authenticate once and thereafter gain access to protected resources and Web sites elsewhere.

This paper presents an overview of a federated identity model. When an SP requests authentication and attributes, the identity selector transmits the set of claims requested about the user inside a digitally signed security token. This set of claims corresponds closely to the notion of a **SAML** assertion, and, in fact, one of the supported token types is a SAML token.

hrsg. von Christoph Meinel, Plattner, H., Döllner, J., Weske, M., Polze, A., Hirschfeld, R., Naumann, F., Giese, H. (2009) Proceedings of the 3rd Ph.D. Retreat of the HPI Research School on Service-oriented Systems Engineering retrieved from http://books.google.ie/books?id=NF9GKbHV30IC&dq=windows+Cardspace&source=gbs_navlinks_s

Cardspace is the new identity management technology that Microsoft introduced after the lessons learnt from the past experiences with .net Passsport. This paper presents Windows Cardspace as an identity metasystem, which allows transaction-based identity. The authors present that it is a component of Microsoft's initiative to create an identity metasystem and that Cardspace rests on a foundation of WS-Security, WS-Trust, WS-SecurityPolicy and WS-

MetadataExchange and will run on any platform that supports those standards. Cardspace is part

of windows Vista and is also included in the .NET framework 3.0. It supports both self-issued

and managed identities, similar to OpenID. Identities are stored as InfoCards in Cardspace and

serve as the digital equivalent to the physical identification cards stored in many people's

wallets. In order to authenticate using a Cardspace managed identity, a Security Token Service

must be used that generates signed, encrypted tokens conforming to the WS-Trust standard.

**Mercuri, M (2007) Beginning Information Cards and Cardspace: From Novice to
Professional** retrieved from http://library.books24x7.com.dml.regis.edu/book/id_25510
/viewer.asp bookid=25510&chunkid=0869018035

This book presents a detailed overview of information cards and Cardspace. It introduces

the concepts and requirements behind identity management introducing the laws of identity and

the identity metasystem.

The book then proceeds to present examples of how to implement Cardspace in a web

application and also how Cardspace interacts with other software technologies such as ASP

.NET2.0 and the Windows Communication Foundation and automating Card issuance with WF.

Microsoft (2010) Identity_Selector_Interoperability_Profile_V1.5_Guide.pdf
retrieved from http://www.microsoft.com/downloads/en/details.aspx?FamilyID=b94817fc-3991-
4dd0-8e85-b73e626f6764&displaylang=en

The Identity Selector Interoperability Profile V1.5 was used to implement the Windows

Cardspace software in Microsoft .NET Framework 3.5 Service Pack 1.

The Information Card Model allows users to manage a portfolio of Digital Identities from various authorities, and employ them in various contexts where they are accepted to access online services. The model embodies the patterns and messages of WS-Trust, but can be implemented using lightweight protocols like HTTP POST as well as the SOAP-based WS protocols. A crucial application for this model is to establish a framework in which consumers of user identities can ask for exactly what they need, and providers of identities can furnish the needed identity with intermediation by the user when appropriate. In the Information Card Model, Digital Identities are encoded as Security Tokens containing claims about a user made by an Identity Provider (IP) and presented to a Relying Party (RP). The Security Token presented may be used for authenticating the user and/or providing authorized access to services offered by the Relying Party. Furthermore, relying parties can express their identity and other security requirements in the form of Security Policy that can be queried by client applications through which the user desires to access the services offered by the Relying Party.

Sharp, J., (2007) Microsoft Windows Communication Foundation Step by Step
retrieved from http://library.books24x7.com.dml.regis.edu/book/id_18611/viewer.asp?
bookid=18611&chunkid=125055753&override=1&tlang=ENUS

This book presents an introduction to the Windows Communication Foundation and presents a chapter on using WCF with Cardspace. There are examples provided for implementing Cardspace with WCF with sample code. There is a good summary of how Cardspace works and there is also a treatment of claims based authentication in a federated environment.

Cavoukian, A . (n.d.)7 laws of identity The case for privacy-embedded laws of identity in the digital age retrieved from http://www.ipc.on.ca/images/Resources/up-7laws_whitepaper.pdf

This paper recognizes and is inspired by the "7 Laws of Identity" formulated on an open blog by a global community of experts through the leadership of Kim Cameron, Chief Identity Architect at Microsoft. The resulting "Identity Big Bang" will hopefully enable the Internet to evolve to the next level of trust and capability.

A universal identity metasystem will also have profound impacts on privacy since the digital identities of people – and the devices associated with them – constitute personal information. Care must be taken that a universal, interoperable identity metasystem does not get distorted and become an infrastructure of universal surveillance.

Cameron, K. (2005) The Laws of Identity retrieved from http://www.identityblog.com/stories/2005/05/13/TheLawsOfIdentity.pdf

The Internet was built without a way to know who and what you are connecting to. This limits what we can do with it and exposes us to growing dangers. If we do nothing, we will face rapidly proliferating episodes of theft and deception which will cumulatively erode public trust in the Internet.

This paper is about how we can prevent that loss of trust and go forward to give Internet users a deep sense of safety, privacy and certainty about who they are relating to in cyberspace. Nothing could be more essential if new Web-based services and applications are to continue to move beyond "cyber publication" and encompass all kinds of interaction and services. Our approach has been to develop a formal understanding of the dynamics causing digital identity

systems to succeed or fail in various contexts, expressed as the Laws of Identity. Taken together,

these laws define a unifying identity metasystem that can offer the Internet the identity layer it so

obviously requires.

Cameron, K. (n.d) Identity Weblog retrieved from <http://www.identityblog.com/>

This weblog is dedicated to the issues surrounding digital identity. It is about building a

multi-centered system of digital identity that its users control. The author , who is is the Chief

Architect of Identity in the Identity and Access Division at Microsoft, where he champions the

emergence of a privacy enhancing Identity Metasystem reaching across technologies, industries,

vendors, continents and cultures, presents the idea that digital identity has a complex relationship

with flesh-and-blood identity, which I'll call natural identity. Sometimes we want digital

identity to correspond to natural identity, and sometimes we want the two to be isolated, or the

knowledge of the connection to be highly controlled. This has become necessary because the

digital world has its own "physics" that is quite different from that of the natural world. Here

space becomes more or less irrelevant and isolation very difficult to achieve, while "now"

extends through great slices of time. The result is not only that our friends and loved ones are

closer: so is every actor, good and bad, and every monitoring device in the world.

O'Neill, M., Hallam-Baker, P., Mac Cann, S., Shema, M., Ed Simon, E., Watters,

P.A. and White, A. Web Services Security retrieved from http://library.books24x7

.com.dml.regis.edu/book/id_6147/viewer.asp?bookid=6147&chunkid=0000000001

This guide explains how to implement secure Web services and includes coverage of trust, confidentiality, cryptography, authentication, authorization, and Kerberos, along with details on SAML, XML, XKMS, HTTP-R and more.

This text introduces XML and then SAML. It also introduces WS-Security and the relationship between WS-Security and SAML.

Haï-Binh LE, Bouzefranei, S. (2008) Identity Management Systems and Interoperability in a Heterogeneous Environment retrieved from

http://www.ieeexplore.ieee.org.libgate.library.nuigalway.ie/stamp/stamp.jsp?tp=&arnumber=476 0564

This paper aims at studying two main solutions to digital identity management, Cardspace from Microsoft and Liberty from Liberty Alliance Project, and the problem of interoperability between them in a heterogeneous environment. It also proposes some recommendations for implementing the interoperability at some given levels.

Malinen, J. (2006) Windows Cardspace retrieved from

http://www.tml.tkk.fi/Publications/C/22/papers/Malinen_final.pdf

This paper presents that Windows Cardspace allows users to manage their digital identities from various identity providers and offers a consistent user interface for controlling what information they are sending about themselves to relying parties. Windows Cardspace is based on SAML tokens and WS-Trust, WS-Security, WS-Policy, and WS-SecurityPolicy standards for managing and requesting these tokens. The protocols and metadata formats used by

Cardspace are not compatible with those used by Liberty, Shibboleth, and SAML, but it supports the security tokens used by these other identity management standards.

McLaughlin L., (2006) What Microsoft's Identity Metasystem Means to Developers retrieved from http://www.ieeexplore.ieee.org.libgate.library.nuigalway.ie/ stamp/stamp.jsp?tp=&arnumber=1576668

This article presents a general discussion on the identity metasystem and the impact it may have on developers. Will it make life easier or harder? Developers must deal with such identity technologies as publickey technologies, Kerberos (used with Active Directory and in some Unix environments), and X.509 (used with smart cards). There's also the Security Assertion Markup Language, an XML standard for authentication and authorization. SAML was developed by the OASIS (Organization for the Advancement of Structured Information Standards, www.oasis-open.org/home/index. php) Security Services Technical Committee, which is tackling the issue of single-sign-on technologies. "This means different sets of APIs for each, and a lot of work supporting all these different models," Shewchuk says. The common specs are known as the WS-* Web services architecture. There's also an encapsulating protocol known as WS-Trust, and negotiation tools called WS-MetadataExchange and WSSecurityPolicy. Microsoft's goal is that new technologies plug into the architecture as they arrive, using the common specs, which it notes are freely available and traveling through open-standards groups.

Moroney L., (2008) Beginning Web Development, Silverlight and ASP.NET AJAX: From Novice to Professional retrieved from http://library.books24x7.com.dml.regis.edu/book/ id_25484/viewer.asp?bookid=25484&chunkid=0815393377

This reference has a chapter on programming with Cardspace in .NET 3.0. It has examples of programming from the client side and also developing a website using Cardspace. It also starts by introducing the setup for the development environment which is a good starting point. Cardspace is also explored from a user point of view.

Lezoray, J.-B. ; Pasquet, M. (2009 Enabling collaboration between heterogeneous circles of trust through innovative identity solutions retrieved from

http://www.ieeexplore.ieee.org.libgate.library.nuigalway.ie/search/srchabstract.jsp?tp=&arnumber=5067517&queryText%3Dweb+services%26searchWithin%3DTrust%26searchWithin%3DCardspace%26openedRefinements%3D*%26searchField%3DSearch+All

This paper presents that during the last number of years, the growth of e-commerce has been considerable due to the increase of user confidence in secure electronic payment. Many web services have been developed and some decided to establish links of confidence, also called circles of trust. A user that accesses a web service in a circle of trust can also access other web services of the circle without additional authentication. In that way, the web services offer is larger and clients' demands more satisfied. Circles of trust are naturally composed of web services from the same type of activity. In this article, we address the problem of federating heterogeneous circles of trust. The main objective is to develop new e-services based on the composition of heterogeneous services. For instance, an application can be the electronic registration of a child to a day-care center: parents will need documents from their bank (in order to pay), from the government (to prove its identity), and from other sources. The preliminary results introduced here are issued from a French innovative research project called FC2 that deals with the federation of heterogeneous circles of trust.

This paper considers information cards and how information cards put the management of the identity back into the hands of the user. Information cards are a metaphor of business cards or credit cards. The user's set of InfoCard is stored on his local computer. Each card represents one of his digital identities, and contains a set of claims describing it (name, date of birth).

OASIS (2009) Identity Metasystem Interoperability Version 1.0 retrieved from

http://docs.oasis-open.org/imi/identity/v1.0/os/identity-1.0-spec-os.html#WSMex

This document contains the specification for Identity Metasystem Interoperability. The Identity Metasystem Interoperability specification prescribes a subset of the mechanisms defined in WS-Trust, WS-Trust , WS-SecurityPolicy, WS-SecurityPolicy, and WS-MetadataExchange to facilitate the integration of Digital Identity into an interoperable token issuance and consumption framework using the Information Card Model. It documents the Web interfaces utilized by browsers and Web applications that utilize the Information Card Model. Finally, it extends WS-Addressing's endpoint reference by providing identity information about the endpoint that can be verified through a variety of security means, such as https or the wealth of WS-Security specifications

.

John R. Vacca (2009) Computer and Information Security Handbook retrieved from

http://library.books24x7.com.dml.regis.edu/book/id_32165/viewer.asp?bookid=32165&chunkid=675435975&override=1&tlang=ENUS

This book presents a chapter on the evolution on identity management - it provides an overview of identity management solutions from Identity 1.0 to Identity 2.0. The first

digital identity appeared when a user was associated with the pair (username, password) or any other shared secret. This method is used for authentication when connecting to an account or a directory. It proves your identity if you follow the guidelines strictly; otherwise there is no proof.

In the context of Web access, the user must enroll for every unrelated service, generally with different user interfaces, and follow diverse policies and protocols. Thus, the user has an inconsistent experience and deals with different identity copies.

In addition, some problems related to privacy have also emerged. Indeed, our privacy was potentially invaded by Web sites. It is clear that sites have a privacy policy, but there is no user control over her own identity. What are the conditions for using these data? How can we improve our privacy? And to what granularity do we allow Web sites to use our data?

Jones, M.B, (2006) A Guide to Supporting Information Cards within Web Applications and Browsers as of Windows Cardspace v1.0 retrieved from

http://msdn.microsoft.com/en-us/library/aa480726.aspx#infocardwebguide_topic1

This article presents an overview of how to implement Cardspace. Two scenarios are presented in this article – the basic authenticating protocol when using an information card to authenticate at a website. This article references the standard Identity Metasystem protocols. The second scenario presented relates to the protocol flow when using an Information Card to authenticate at a Web site, where the Web site employs a relying party STS.

An example of invoking an identity selector from a web page is presented with a description of the optional invocation parameters presented.

WS-Trust 1.3 retrieved from   http://docs.oasis-open.org/ws-sx/ws-trust/200512/ws-trust-1.3-os.html

This specification defines extensions that build on to provide a framework for requesting and issuing security tokens, and to broker trust relationships. WS-Security defines the basic mechanisms for providing secure messaging. This specification uses these base mechanisms and defines additional primitives and extensions for security token exchange to enable the issuance and dissemination of credentials within different trust domains.

Novák I., Velvárt A., Granicz A., Balássy G., Hajdrik A., Sellers M., Hillar G.C., Molnár A, Kanjilal J. (2010) Visual Studio 2010 and .NET 4 Six-in-One retrieved from http://library.books24x7.com.dml.regis.edu/book/id_34981/viewer.asp?bookid=34981&chunkid=861839446

This book presents a history of Visual Studio – its worth knowing where it started and how's it been evolving. The roots of Visual Studio go back for almost 19 years, back to the point somewhere between the release of Windows 3.0 and 3.1. A chapter is dedicated to the history and evolution of the .NET framework. Before the age of the .NET Framework, the traditional programming language of Windows was C for a long time. Since 1998, more and more programmers have started to use Visual Basic (VB), primarily because of the release of Visual Studio 6.0. VB was very popular immediately from the beginning — thanks to its simplicity of creating complex user interfaces (UIs), COM/COM+ server components, and the accessing of data. Microsoft released the first version of the .NET Framework in the middle of the year 2000, but the history of the .NET Framework goes back to the late 1990s. During the planning of the .NET Framework, software engineers became very excited because they wanted to create a brand

new developer framework with a brand new approach, without the deficiencies and problems of

the old platforms.

Rits, M. Rahaman, M.A. (2006) Secure SOAP Requests in Enterprise SOA retrieved

from http://portal.acm.org.dml.regis.edu/citation.cfm?id= 1180367.1180382&coll=

DL&dl=GUIDE&CFID=12387947&CFTOKEN=31250318

In this paper the authors present that "SOAP message exchange is one of the core

services required for system integration in Service Orientated Architecture (SOA)

environments." The security of these messages is question in relation to XML rewriting attacks.

Related work has been perform in this are in relation to preventing these types of attacks through

the use of a SOAP account – this paper concerns itself with an attack on the SOAP account itself.