2

Hints Toward a Solution

Now that we understand how today's security schemas work and how they evolved to their current state, we realize the reasons why they fall short in providing a common identity layer for the entire Internet. It is time to put into practice the lessons learned and devise a long-term solution, finally immune from the errors and shortcomings that afflict today's patchwork of partial solutions.

The section "A World Without a Center" stresses the reasons why a universal identity layer didn't spontaneously emerge to date and highlights that a truly sustainable solution must address the needs of all the disparate parties that have an interest in the Internet.

The section "The Seven Laws of Identity" describes the choral effort that the industry poured into determining the mandatory requirements that must be met by any acceptable solution to the online identity problem. The seven laws of identities are a compact formulation of those findings.

The section "The Identity Metasystem" presents a model for describing roles, transactions, and relationships of systems in which identity information is exchanged. The section explores the expressive power of the Identity Metasystem and its soundness, describing how its various parts can be composed for handling different example scenarios in ways that are fully respectful of the identity laws.

The section "WS-* Web Services Specifications: The Reification of the Identity Metasystem" provides a brief overview of the advanced web services specifications, positioning the trend in the industry landscape and delving into the details of some especially relevant specifications. After all the pertinent details have been spelled out, the text shows how the abstract constructs in the Identity Metasystem find a concrete counterpart in the web services world. A sustainable solution for the online identification system has finally been found, and the technological means to put it into practice are already mainstream.

The section "Presenting Windows CardSpace" positions Windows CardSpace in the Identity Metasystem, explaining its role and its relationship to the other components of the solution.

By the end of this chapter, you will understand the Identity Metasystem, how it works, why it is the way it is, why it can aspire to be a global solution, and why former attempts fell short. The Identity Metasystem is the ecology in which Windows CardSpace is designed to thrive. Gaining a solid understanding of the model is the best way to learn how to take advantage of this new technology.

A World Without a Center

HTTP is the collagen of the Internet

The fabric that keeps the Internet together is fairly simple from a technical standpoint. You saw in the preceding chapter how the content-publishing infrastructure (browser plus web server plus HyperText Transfer Protocol [HTTP]) proved flexible enough to be twisted in the wide gamut of online applications we see today. You have also seen that security concerns, specifically about identity, are a serious seatback for the activities involving high-value transactions. The technology for addressing those concerns, or at least significantly mitigating them, already exists. We took the time to understand strengths and inadequacies of the main authentication schemes, and it's clear that cryptography and token-based schemas have the potential to provide a technical solution to the problem. In fact, for the most part, the problem is not technical at all.

The Internet is a means to many ends

The reality is that the Internet is just an enabling infrastructure. It is the stage to an incredible number of different dramas, all involving different actors with their own agendas. Every service provider runs his or her interests on the Internet for his or her own reasons, according to his or her own business model and practices; and unpredictable new business models thrive and decline at stunning pace without central supervision or governance of sort. (At the time of this writing, the huge success of twitter.com is baffling old-school analysts.) The concept of identity plays a key role in every service or activity that provides or manipulates value. It should not come as a surprise that every business wants to exercise control over the way in which identity is managed for their assets so that they can ensure that it is inline with their business goals. Different businesses will have different expectations from identity management. An enterprise giving remote access to its employees will want to make sure that access levels are enforced, striking the delicate balance between ease of access and security. The same enterprises, when offering online services to customers, will have a different agenda. Customers will need to be authenticated with the right security assurances, sure, but the highest-order bit will be how to capitalize on relationships, retain customers, achieve loyalty and prevent departures, leverage customer profiles for improving sales or selling info to marketing firms, handle privacy and regulation concerns, keep user-profile data fresh, and many other considerations. Those are all business goals that can deeply affect how customer identity is handled from the technical standpoint; furthermore, any operator will give different weights according to the kind of service they provide. Just think of the use that Amazon.com would make of its user profiles, as opposed to matchmaker businesses such as eHarmony.com. That's not all. As the usage of new technologies rises in government functions and practices (the so-called **eGovernment**), institutions expose more and more of their operations to online consumption. Their view of identity is influenced by the existing relationship they have with citizens, and the assurances they have to provide must be inline with the official function they are called on to accomplish.

Controlling how identity is managed is appealing to many

The different ways in which identity is defined, exchanged, and manipulated in a certain transaction defines a context. As mentioned previously, everybody has a strong interest in controlling the identity context in his or her transactions. For that reason, the absence of a constraining standard is exactly what allows businesses to adopt their own solutions. Chapter 1, "The Problem," is full of examples of those identity one-offs. The Internet does not have an identity layer, and this is one of the key reasons behind all the problems we have with authentication today. But if the Internet *did* have a native identity layer, and it was not expressive enough for allowing businesses to enforce their requirements, it would be reasonable to expect the rise of proprietary alternatives. Back to square one.

The different views on what identity is or what an identity layer should do are the reason why a common solution didn't spontaneously arise, and it is not plausible to expect this to happen anytime soon. Perhaps more important, that is also an indication of what a universal identity layer should look like. It will need to have enough expressive power so that present and future businesses will be able to use it according to their needs; otherwise, it will face the same fate of existing schemes.

At the end of the day, it is user acceptance that makes or breaks the system

Although services providers are a very important part of the equation, they are not the entire story. User acceptance makes for the success or the failure of many online services. Systems have to walk a thin line between ease of use and security assurances offered; context information considerations, such as how private is the data being exchanged at the moment, are powerful influencing factors for pulling opinions on one side or the other of that line. We have seen in Chapter 1, in the sections "Passwords: Ascent and Decline" and "The Babel of Web User Interfaces," how users have trained to cope with inefficient and insecure systems. The consequences of those shortcomings are often felt at moments apparently unrelated to the authentication experience, such as when you spot an unauthorized purchase days after the last home-banking transaction. Hence, the user is not always able to recognize the causal link between aspects of a bad authentication system and the issues it causes. Add this to the difficulty the user has when trying to figure out what is going on during a transaction (such as whether the website rendered in the browser is truly the intended one). This is another facet of the problem that a common identity layer has to solve. It has to offer a user experience that is acceptable, and at the same time it has to protect the user interests without getting in the way.

The Internet does not have a center

The Internet does not have a center. This claim can be supported from many points of view: no common governance, many service providers with different agendas, and a mind-boggling number of users who often defy attempts to partition and classify them. All of those entities want a say about how identities are managed, and rightfully so. Any truly sustainable solution must address their concerns. That is the minimum bar for entertaining any hope of a strategic solution to the problem.

The Seven Laws of Identity

As we have seen consistently in Chapter 1, a common mistake in the evolution of the IT industry has been trying to extend the use of existing technologies "as is" to deal with problems that are only apparently similar to the ones the original technology was meant to solve.

A strategic solution often requires restarting with a blank slate

Breaking this impasse requires distancing ourselves from the tools we (maybe erroneously) believe we may use for solving the problem and trying to consider just the problem itself. By doing so, we might not get an instant solution, but we can certainly obtain precious and unbiased insights into what an acceptable solution may look like. At least as important, we can also learn to recognize nonsolutions. By understanding what the properties are we can't do without, we gain a valuable compass for navigating the problem space toward a solution.

Cameron ignited the debate about identities with a public blog

Kim Cameron, an architect at Microsoft, tried to do exactly that. In 2004, he created a blog, www.identityblog.com, from which he elicited discussions on identity management. The focus was on understanding what worked and what didn't in current and past identity management efforts, with an accent on understanding the deep reasons for why things went one way or the other. Issues were examined from multiple angles: technology, social considerations, usability, and privacy. Vendor differences were suspended in the name of understanding the problem from a broad industry perspective. No topic was off limits; in fact, one of the most studied topics was the shortcomings of the most ambitious universal authentication scheme attempt at the time, Microsoft Passport. Cameron successfully involved key industry players and thought leaders from the entire community in the dialogue, gaining consensus even from the least expected sources, such as prominent figures in the open source world.

In 2005, Cameron distilled the results of the discussions in a single white paper, "The Laws of Identity," where the main findings are summarized in concise format. The white paper lists seven "laws." They are principles to which, according to the previously mentioned investigations, an identity management system must comply to be viable. Since the white paper's publication, the identity laws have become immensely popular and are considered by many the manifest of the new user-centered identity management movement. The seven laws, listed in their concise form, are as follows:

The "Laws of Identity" white paper summarizes the findings of an open, industrywide conversation

1. User Control and Consent
2. Minimal Disclosure for a Constrained Use
3. Justifiable Parties
4. Directed Identity
5. Pluralism of Operators and Technologies
6. Human Integration
7. Consistent Experience Across Contexts

The identity laws are not dogmatic by any measure, nor are they blindly prescriptive. Ultimately, they are a set of sound and pragmatic principles, derived from real-world experience, that anybody can verify at any given moment. Their goal is to give rise to a system that can enjoy true acceptance while serving the intended purpose of an identity system to the full satisfaction of all the parties involved. The seven identity laws define how to successfully extend the Internet with an identity management layer. In the remainder of this section, we examine the laws one by one.

The laws of identity are not dogmas. They derive from very practical considerations

In the following section, "The Identity Metasystem," we describe a solution that abides by such laws. The Identity Metasystem is the model of reference for which Windows CardSpace has been designed.

The Seven Laws of Identity and the Four Tenets of Service Orientation

If you are familiar with service orientation, here is an analogy for you.

To some extent, the seven identity laws are similar to the four tenets of service orientation. The tenet "Share schema, not object" is not an absolute dogma, and no service-orientation police will come to arrest you if you don't respect it. The consequence, however, is that you will not be able to serve loosely coupled clients that don't understand your object technology. It is observing exactly this shortcoming that brought on the formulation of the tenet in the first place. Similarly, if you develop an authentication schema that mandates one specific technology, you are breaking law 5 (see "Pluralism of Operators and Technologies"). Nobody among the authors of this book will come to haunt you for that, but you should be aware of the fact that your authentication schema may have shortcomings in certain areas and is probably not fit to become the universal identity management system.

User Control and Consent

Technical identity systems must only reveal information identifying a user with the user's consent.

—*The Laws of Identity*, Cameron, 2005

This is truly the most fundamental principle of an identity management system.

The user must always understand what is going on

The user must be able to decide to whom he discloses information, which specific data is being shared, when exchanges take place, what the purpose is for which the information is gathered in the first place, and what the trail is that a specific transaction may leave behind. To make that degree of control even possible, the user must understand what is going on. Always.

In today's practices, we witness gross violations of the first law everywhere. Remember the concept of server authentication, discussed in the sections "The Babel of Cryptography" and "The Babel of Web User Interfaces" in Chapter 1? The lousy job we do today of making users able to understand to whom they are disclosing information is one of the root causes of phishing, which is by itself one of the main causes in the decline of the use of the Internet for high-value transactions. A violation of the first law of this magnitude promptly leads to diminished acceptance.

Today the user is often not in control. The consequences are serious

There are other somewhat subtler violations to consider. We are used to the idea that what we transfer in an authentication transaction is just the credentials so that we can unlock our identity on the service provider. In fact, there are many occasions in which our identity can flow from one service to the other. In Chapter 1, in the section "HTTPS, Authentication, and Digital Identity," we have a real-world example in which frequent-flyer privileges of a customer are shared between two commercial partners. In the sections "Hard Tokens" and "Issued Token–Based Authentication Schemes" you saw technologies that give to identities a vessel for traveling across different entities, such as the Security Assertion Markup Language (SAML) token representing the assertion, "Alice is a principal in my realm, and she just successfully logged in using username/password as credentials," mentioned in the section "SAML." This covers the feasibility of the operation from the technical standpoint but says nothing about the way in which what is happening surfaces to the user's attention. Let's say that you are working for an important technology company that has a close partnership with a hardware provider. By virtue of that partnership, purchasers at the hardware vendor site enjoy automatic deals applied specifically for your company. The experience is seamless. While you are browsing your corporate intranet, you click a link to the hardware vendor, and the web store automatically recognizes you as an employee of a partner company; you get a welcome banner with your name, and the deals on the

Even single sign-on systems may hide violations of the user in control principle page are adjusted accordingly. That's the magic of single sign-on (SSO; see the section "SAML" in Chapter 1). Sometimes the transition may be so seamless (thanks to layout customizations) that you might not even realize that you are now in a different place and that an authentication step has been performed at all. That might be very convenient from the usability standpoint, but you can't say you had much control over the information about you that flew from your company to the hardware vendor website. From what you can see, the partner website was able to determine your name and your status of employee. But what if much more information was transmitted without your knowledge or consent? If the hardware vendor acquires information about your salary or your home address, something that typically you would not want to disclose, consequences vary from targeting according to the advertisement on the web store to selling that information to marketers, junk mailers, or worse, burglars. Wouldn't it be much better to be warned that your identity is about to be disclosed and to whom and what information is specifically being requested? Wouldn't you require, after you realize what is going on, a mechanism for opting out if you feel it is risky?

That's the essence of the first law. Knowledge is power. Awareness of the situation brings the ability to take action responsibility, which in turn brings confidence and the feeling of being in control.

Minimal Disclosure for a Constrained Use

> *The solution which discloses the least amount of identifying information and best limits its use is the most stable long term solution.*
>
> —*The Laws of Identity*, Cameron, 2005

Let's focus once more on the partnership example we introduced in last section "User Control and Consent."

The "need-to-know basis" principle applies to identity

Your company negotiated access to the hardware vendor website to fulfill a business need, empowering employees to purchase devices for the company with a process as agile as possible. The purchase process needs to gather some specific data from every shopping session. The fact that you are an employee of a certain partner, your name, the business address to which items will have to be shipped, coordinates for emitting an invoice, the spending limit that has been assigned to you or to your role. Omit any of those data, and the transaction cannot take place. Do they need to know your salary? Your home address? Your blood type? You religious beliefs? Your hair length? They would probably be happy to have some of that information, but the answer to all these questions is a resounding no. The reason for which you are shopping at their website is performing purchases for your employer. The fact that you are a geek and that later that night you will buy an oscilloscope for your personal enjoyment is not relevant now, and therefore your home address should not be part of the current transaction.

Even if the hardware partner is acting in good faith and does not sell your personal data to junk mailers, disclosing more data than necessary is still a very bad idea. A rich archive of personal details is a treasure trove for identity rogues and makes the company a very palatable target of attacks. The liability is also higher in case of accidents. A laptop forgotten on a train with a list of names plus company addresses is much less likely to unleash a class action lawsuit than the same list of names with home addresses, birth dates, and so on.

Incorrect disclosure of data can have negative effects even a long time after the event occurred

The principle of minimal disclosure can and should also be applied at a finer level of granularity. A business selling wine, in a country where alcohol consumption is allowed only after a certain age, may be tempted to store the birth date of recurrent customers. That is a point of liability that could be easily avoided because it is possible to store only the aspect relevant to the business (that is, a Boolean expressing if the customer is above or below the threshold age).

A negative example: How the Social Security Number is handled in the United States

Unfortunately, today's identity silos often invite practices in open violation of the second law. Many business operations in the United States require disclosure of the Social Security Number or SSN (see the sidebar "America and Identity Theft" in Chapter 1). It often happens that the SSN will end up being memorized in the user profile, even if there's no need to know it beyond the current transaction. It is kept just in case because it is information difficult to obtain. In the most appalling cases, it is even misused as record key because it is a unique identifier. The latter are the worst cases. Not only is the SSN very valuable information per se, it also provides a key for aggregating and interpreting identity data stolen elsewhere! That means spreading the damage across different identity contexts, annihilating one of the only advantages of today's identity silos. Because it is so difficult for information to flow between silos, the scope of damage is often contained too.

The principle of minimal disclosure for constrained use is very pragmatic, and the strategic value of the practice is clear. It is clearly proven architectural wisdom applied to the context of identity.

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 Laws of Identity*, Cameron, 2005

One of the first adopters of Microsoft Passport was Victoria's Secret. At the time, it was not a well-known brand in Italy. When one of the authors found out that it was a lingerie brand, he was puzzled. He spent a good deal of time trying to understand the business reasons for which Microsoft needed to be informed of the details of his Valentine's day purchases.

Understanding the circumstances requires recalling what the Internet was in the few years after Y2K. Today it is almost unthinkable for any company not to have substantial web presence. In 2001, there were still many important companies without a website, and the bursting of the dotcom bubble had scared the industry enough that they backed off any mainstream strategy related to the Web. Brick-and-mortar companies often didn't have investments in or know how to invest in web properties: Website creation and maintenance were massively outsourced, almost as experiments and PR bangs, every move clearly giving away that the energies were still on the traditional channels. The Web was not as ubiquitous as today. The demographics of habitual customers, the main target, were not expected to overlap much, from the very beginning, with those of the audience of the website. Web-based campaigns were far from today's maturity in term of tools, demand, structured offerings, and raw material (read, eyeballs).

Internet presence wasn't always considered a strategic asset

In that atmosphere, it should not come as a surprise that somebody saw authentication just as another "feature" of the website, and as such suitable to be handled by third parties, too.

The Passport offering was very convenient because it relieved sites from the hassle of managing their own authentication infrastructure, a very delicate aspect of the website architecture. That was the intended role of Passport in the purchase of a Valentine's day present; Microsoft was just an infrastructure provider.

Passport was designed as a turnkey system

As the Internet became what it is today, many of the conditions that made authentication outsourcing appealing started to fade. It became unmistakably clear that the web presence *is* a strategic asset, while at the same time online activities became more complex and feature-rich. The attention and resources devoted to it by companies increased. As the number of Internet surfers

Companies realized that a turnkey system was not always suitable for their interests grew an order of magnitude, the importance of the Web as a medium for reaching customers grew, too. Any information about the user became precious for maintaining loyalty, predicting behavior, and targeting offerings. Online advertising exploded. It was like the offline world, but the eyeball economy made everything faster and global reaching. In these new conditions, in which somebody can earn revenue just by having you look at one page, outsourcing identity management just does not make business sense. That's why nowadays we are so surprised at the attempt to extend the Passport authentication scheme beyond Microsoft assets, but at the time there was some reasoning behind it. In fact, other big Internet players are betting on similar systems still today while Microsoft endorses the Identity Metasystem (see the section with the same name).

While online business went through all those transformations, maturity and awareness in the usage of the Internet increased. Once past the convenience of remembering just a single set of credentials, users and operators began to realize that the web farm of one single operator was in the position of keeping track of all their movements and didn't like the idea. When the technical reasons for outsourcing authentication disappeared, or were greatly reduced, there was no justification for that situation. If you add that some websites tried to make it as unobvious as possible that they were in fact relying on Passport, you can see how users didn't feel much in control.

The presence of a party in a transaction must be justifiable to the eyes of the user

In fact, "Justifiable Parties" is another flavor of the "User Control and Consent" law. Every time the user discloses his identity information, he needs to be able to assess not only to whom he is sending data, but also understand its role in the current transaction and the implications of its involvement. Let's get back to the wine seller example we introduced in the previous section. The merchant needs to know whether you are of age before serving you alcohol, and he may not take your word for it. In the offline world, the natural solution entails extracting your government-issued ID document and exhibiting it. As we have seen in Chapter 1, in the section "Hard Tokens," this is an action that more and more often we can metaphorically perform in the digital world, too. Here the reasons why the government is involved in the transaction are obvious. The merchant needs to know whether I am of age and won't take my word for it. However, he is willing to believe what the government says about me. Short of finding another entity that the merchant trusts, if I want to go on with the transaction I have no choice but to accept government involvement. (Notice that I still must be given the choice of opting out, when I learn the merchant's policy). Again, there are finer points to be made. The user is the ultimate judge of the justifiability of the participation of somebody in a transaction, and all information for making that call must be made available. Consider this. What if every time you use your electronic ID, your government keeps track of with whom you are conducting business? Would you still say that government involvement is justified? It probably depends. Somebody will recognize that this is a necessary security measure if the transaction is applying for a visa with a foreign government, but it is plain abuse to keep record of how many times you buy wine in a month; somebody else will be okay with both; and so on. This is just one among many examples. When was the last time that a marketing company asked for your permission for monitoring your buying habits? The point is that it is the user who should be the one who justifies the terms of the participation of one entity in the transaction, and a good identity schema should do everything for facilitating that judgment call. That means explicitly and clearly communicating policies about information usage.

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 Laws of Identity*, Cameron, 2005

The fourth law further refines the concept we have of digital identity.

The intended audience is what defines the "direction" of an identity

In Chapter 1, we debated the problem of server authentication, and we hinted how Public Key Infrastructure (PKI), certificates and Secure HyperText Transfer Protocol (HTTPS) can help in pinpointing the identity of websites. Who is the beneficiary of that help? In the case of a public website, it will be the "public" itself. Everybody that is not the website itself or, as Kim solipsistically put it in the white paper, "all the other identities."

An omnidirectional identity defines the public identity of an entity

We call that kind of identity **omnidirectional**. It is an identity meant to be understood by everybody. This identity will contain the info necessary for the public to decide if they want to do business with it. X.509 certificates and associated URLs are the most natural example in this context, but the instances in the offline world abound. You may have seen at some conferences those badges that display the attendee name, the company he or she is affiliated with, his or her role in the conference (attendee, speaker, staff), and the languages he or she can speak. That information is beamed to everybody coming within visual range of the badge and helps everyone else to recognize the bearer and the methods of interaction. The Web 2.0 breeze that blows on the Internet these days brings many means of doing the same thing online. For example, at the time of writing, Opinity (www.opinity.com) offers to its users a unique URL that provides the function of omnidirectional identifier. (The Opinity URL for Vittorio is http://vibro.opinity.com.)

A unidirectional identity defines the identity of an entity in the limited scope of a transaction

When an individual enters a transaction, however, the identity he uses is unidirectional. That is, the identity transmitted is meant only to identify the user with the service provider currently engaged. If you are buying an airplane ticket on one website and booking a hotel room on another, the authentication scheme should not help the two websites to join their data and understand that you are the same person (and afterward send you advertisements about shuttle services between your destination airport and your hotel).

An identity management schema should not provide means for correlating identities across different contexts

This is a very subtle point. A typical objection at this point is this: What if both sites require name and birth date? What can an authentication system do to prevent the two businesses from joining data together? The answer to that is, not much. If the two businesses require name and birth date to perform their function, there's nothing that can be done. You might require that data be encrypted with the public key associated with each site so that the data is not mutually visible, but that covers just the transmission. As soon as the information arrives at its intended destination, two dishonest service providers can still share profiles and search for a match. That's one of the reasons why using something unique and personal such as the SSN is really, really bad practice. The point of the Directed Identity law is that such a possibility should not be offered by the identity management schema in itself. In other words, an authentication schema should not rely on mechanisms that could give rise to correlation handles. Imagine a situation in which the services you are using require you to sign in, but they do not require any further information about you besides the credentials you use for authenticating. One example of such a service could be a photo-retouching website. After having signed in, you can upload one picture, and somebody will fix red eyes on-the-fly and send it back to you in the context of the same session. Another such a service could be a traffic information service or weather reports. When you sign in, you can get information about one area of choice. For both services, you are just sending the credentials required to verify that you subscribed to the service. In that case, an authentication schema respectful of the directional identity law will not allow the traffic service to realize that the person who asked about the situation on Highway 90 is actually the same person who sent those "oh so weird" pictures to be retouched. That separation will typically be obtained by the identity management scheme by ensuring that no two websites share the same identifier for the same user. But that's just an implementation detail. What counts is that the scheme does not enable the kind of abuses previously described; how it accomplishes that does not really matter.

Pluralism of Operators and Technologies

A universal identity system must channel and enable the inter-working of multiple identity technologies run by multiple identity providers

—*The Laws of Identity*, Cameron, 2005

We devoted a good part of Chapter 1 to describing different ways of handling authentication: certificates, SAML, and even passwords. Proposing a single authentication scheme for the Internet has been attempted, but it has failed. As the next lines will hopefully clarify, such an effort is doomed from the very start.

Diversity and variety are inherent in the problem of Internet authentication

We have seen how the features of different systems are the result of the diverse requirements imposed by the contexts in which they are meant to operate. We should not expect those differences to go away, in much the same way as we should not expect that hammers and screwdrivers will eventually converge into one single tool. Furthermore, we have seen how today's scenarios and associated requirements greatly differ from yesterday's. By induction, we can safely assume that the future will pose challenges that we are unable to predict, and hence the solutions will also take forms we cannot foresee today.

People and businesses will have their own preferences and inclinations, and those will be reflected in their technology choices. As the value of the transaction rises, the level of security required will follow suit; different businesses will deal with risk in different ways, formulating their policies accordingly. Different users will have different degrees of tolerance for information disclosure; the concept of what is or is not acceptable in terms of safeguarding one's own privacy will vary widely by communities, cultures, or who knows what other factors. Just think of the example we made in the section "Justifiable Parties" concerning government tracking of electronic ID usage. Some will accept this unconditionally, and some will push back so hard that merchants will have to adopt different technologies for meeting user's privacy demands to remain in business.

Inclusiveness and tolerance are key factors for the success of a global solution

Handling such a diverse mix of tendencies *requires* pluralism of operator offerings and technologies available. An identity management scheme that aspires to be the universal authentication system cannot fail to take the situation into consideration. Embracing and accommodating existing and future technologies is the only way to achieve the goal.

In the section "The Identity Metasystem" we describe a natural solution to the dilemma.

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.*
>
> —*The Laws of Identity*, Cameron, 2005

Chapter 1, and specifically the section "The Babel of Web User Interfaces," described the inadequacies of current practices in making the user understand what is going on during the authentication process. We have seen how the certificates, although perfectly sound from the purely cryptographic standpoint, are not really helping the user to deal with the server authentication problem.

We have also seen how the wide gamut of different user experiences, despite the fact that in the vast majority of cases they all account for the task of entering username and password, confuses the user to the point of making him vulnerable to the simplest phishing attacks.

What works for machines may not work for humans

If we analyze from the pure engineering standpoint the communication sequence when authenticating to a website, we discover an almost universal pattern. Until the communication happens between machines or software entities, the protocols are predetermined and rigidly followed. Every phase mandates message formats and sequences, and the semantic of every step is unambiguously determined. A good example of this point is given in Chapter 1, in the section "SSL Client Authentication." As soon as human intervention is required, however, things change. Even if the task is almost invariably to enter password credentials, every website will implement the functionality in different ways. There is the diffuse idea that the user will "figure it out," so a reasonable set of controls and a sound process behind it will do. The flaw in that reasoning lies in the fact that *reasonable* and *sound* are ill-defined. Apart from the fact that often those systems are designed by computer scientists, who abide by a very different definition of *reasonable* than end users, the entire idea of relying on the user's ability to "figure it out" is extremely dangerous. When the user is expected to recognize to whom he is disclosing his personal data or which kind of information will be sent, the margin for interpretation should be reduced to an absolute minimum. The way of achieving this is planning for human integration, devising interaction mechanisms that properly account for the user capabilities, eliminating ambiguity, and reducing the room for misinterpretations. In other words, when the user deals with identity management matters, he should be constrained by a protocol, too.

Humans can follow protocols, too

Following a protocol is not exclusive to machines. Humans can do it, too, and have done so since forever, every time it is important to have predictable results. We follow a protocol on election day when we go to vote, when we clear a security checkpoint at the airport, when we sign a contract, when the fire alarm goes off in our office building, when we operate a nuclear plant, when we document a process in the context of ISO9000, when we apply for an immigrant visa. The list can go on and on. In those cases, we follow a protocol because there's a lot at stake in terms of risk or resources and, as painful and uninspiring as it may sometimes be, we accept that as a fact of life.

The way in which a universal identity system (please ignore for the time being the term *metasystem* in the law enunciate) should integrate humans is by maximizing comprehension while minimizing ambiguity. That is, a universal identity system should make everything as understandable and incontrovertible as it can be. That implies representing facts and entities in ways that the modern science of human computer interaction deems appropriate and defining rigorously the actions that users can perform and their exact semantics. Clarity claims its price on freedom. A system easy to understand and with fixed semantics will limit the room for creativity. However, when operating a nuclear plant, creativity should not be the higher-order bit. The same goes for making all the users understand whether the information they are being requested to send will travel in the clear through an untrusted network or whether it will be encrypted.

A universal identity system should make everything as understandable and incontrovertible as it can be

Note that this by no means implies limitations on specific authentication technologies. It just states that a universal identity management system should properly accommodate human integration but gives no indications of the architectural layer at which such integration should take place.

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.

—*The Laws of Identity*, Cameron, 2005

While using the Internet, we project our identities all the time; we just don't always realize when we do it. In fact, many users do not actually have a clear picture of what identities they have and how they are used across the various services they make use of. The current user experience in that space is so broken that talking about consistency is difficult. Users do not even have a clear perception of what a security context is by now.

The user might not even have a mental symbol for "digital identity"

Think about it for a moment. The typical user will have a handful of password credentials he uses and reuses (see the section "Decline" in Chapter 1). The actual identities of the user are the sets of relevant facts that are kept on the service provider stores and are unlocked by transmitting the correct set of credentials (see the concept of hostage identity in the section "HTTPS, Authentication, and Digital Identity," in Chapter 1). If a username-password couple is reused across two different services, it will likely correspond to two different identities; this is supremely confusing for the user, who manipulated directly just the credentials and is only vaguely conscious (if at all) of the existence of the associated identities unlocked on the service-provider side. Password manager utilities do not really help, and sometimes they make things worse. By showing that the same username is used across different websites, they may induce the user to believe that he is using the same identity across the group even though the user profiles kept on different service providers may be dramatically different. That is certainly a setback in the attempt to instill context awareness in the user.

Shifting the perspective of the user toward thinking about identity would simplify many operations that are today prone to errors

This last thought experiment describes just what happens at authentication time. However, there are countless other times at which online applications ask you to disclose fragments of our identities. This typically happens when you engage in a high-value transaction, when the service provider needs to reach beyond the online world and gather data from your offline identity. If you are having something shipped, you need to provide your address; if you are handling some administrative practices with your government, you may have to provide your ID number; if you are verifying the status of your immigrant petition, you have to provide your application number; if you are buying something online, you may have to disclose details of the relationship you have with your credit card provider (that is, enter your credit card number). All those things happen in completely different contexts, following different processes, requiring different interaction patterns. The concept of identity, which would be so useful and the natural tool for modeling those transactions, is implicit at best and is more often than not just an emergent property of the system. No wonder that the user has a hard time handling his or her identities effectively! It is like trying to understand the paths that planets follow in the night sky without knowing that the Earth itself spins and everything revolves around the Sun. Without the latter information, those paths are extremely difficult to understand and predict. Adopting the new perspective, however, makes everything crystal clear.

If we want to solve the problem of identity management for good, we need to be like Galileo and rebuild the system on the basis of the fundamental identity mechanics we have discovered so far. That will lead to a more natural and effective way for users to think about identity. Proficiency in managing it will follow suit.

Again, control is key. The user must be aware of the concept of identity

The first thing we can do is make the concept of identity explicit for users. A user should be able to think about his or her identities as clearly as he or she thinks about his or her files, documents and any other abstract entity that has a visual representation in a user experience.

Once the identities are explicitly represented, we have made an enormous step forward. Users can now create identities for all the contexts and the hats they wear: identities for web mail and low-value services, identities as employees of a certain company, identities as citizens of a certain country, identities as members of a dating service, identities as alumni of a certain university, and identities as just about any kind of digital persona they expect to use in their activities. Information will naturally fall into the right place. How much you paid for taxes last year will be in the citizen identity and not in the alumni identity, whereas for your grade point average it will be vice versa.

Consistency across contexts can provide the user with the landmarks necessary for understanding how the identity flows

Once the information is packaged in explicit representations of identities, it is finally possible to reach a level of consistency in all the transactions involving disclosure of identity data. Users can choose which identities are most suitable in every given context, while services can help users to understand the context by explicitly limiting the set of identities they are willing to accept. A service asking for our email and a service requiring knowing our yearly net income can now do so using the same user experience, relying on the new awareness that the user has about his identities. The system must be secure, and the differences between the two requests must be completely clear (see the section "User Control and Consent" and discussion about unambiguous operations in the section "Human Integration"), but the semantic of the two operations is the same. Disclose some part of one of your identities. It makes sense that the user experience is the same, too, just as the procedure for copying a file between two folders doesn't change regardless of whether the file content is the script of the movie *Borat* or the true recipe of the philosopher's stone.

The Identity Metasystem

We can now count on the laws of identity as guidance and as a powerful tool for evaluating whether a solution is truly fit for the task. The time has finally come to unveil a comprehensive, long-term solution to the problems we have described so far.

Of all the seven laws, many are actually just good architectural common sense. The fact that there was the need for a law to be formulated reflects the fact that the Internet grew like a coral reef, without an architect, and things just happened on their own. The tools for solving many of those problems are available and are successfully used in other areas; it's just a matter of engineering them in the identity space.

One law, however, is problematic: the "Pluralism of Operators and Technologies." We have made clear throughout the entire book that diversity is an important and a noneliminable component of the Internet ecology. How can we convince that all entities in operation, today and tomorrow, would abandon their current systems and adopt a new one? Would we even want to do such a thing?

Fortunately, we don't need to. We can create a system of systems, or Metasystem, that will embrace existing technologies and facilitate the dialog among them.

We solve the pluralism dilemma by adding a level of abstraction. A system of systems can embrace new and existing systems

Managing identity entails manipulating common abstract principles, performing specific actions and covering canonical roles. Those are concepts that exist in complete independence of the specific features of the existing and imaginable authentication schemes. Just think of the descriptions we gave of SAML, Kerberos, Secure Sockets Layer (SSL) client authentication and others in the section "The Babel" in Chapter 1. There are important differences in the way they operate, but you can see that there are analogous concepts (such as the idea of token) and messages with the same semantic (such as obtaining a token from an authority).

The Metasystem abstracts away concepts that are common to all identity systems but which are often implemented differently

We can conceive an Identity Metasystem that defines concepts and operations universally valid in the identity space, without bothering about the implementation details; we can devise an integration layer through which the peculiarities of specific identity systems are abstracted out and mapped to and from those generic constructs. Not having an implementation on its own, the Identity Metasystem does not aim to substitute for existing systems. It actually needs them because they provide the implementation fabric it lacks.

A system of systems protects the investment already committed to in existing technologies

This solution enables applying what we have learned about correct identity management without compromises because we don't have to worry about legacy features we need to maintain for the sake of technology. At the same time, the solution invites present and future technologies to participate. By concentrating on the fundamental principles of identity management and leaving the details to single solutions, it is impervious to technological dependencies that would limit its scope and its expected life.

This simple idea has beautiful implications and defines the very physical laws of the universe in which Windows CardSpace is meant to operate. The remainder of the current section describes in depth the Identity Metasystem, introducing many concepts and ideas that are key to understanding the technology.

The section "Some Definitions" formalizes some of the terms used in the context of the Identity Metasystem.

The section "Roles in the Identity Metasystem" describes the essence of the solution by introducing the entities and the relationships that keep together the ecology defined by the Identity Metasystem. The discussion will still be at the model level, without strong references to the actual reification of the solution in today's technology landscape.

The section "Components of the Identity Metasystem" digs deeper into the requirements that must be satisfied for making one Metasystem possible. It will also lay the foundation for the section "WS-* Implementation of the Identity Metasystem" later in the chapter.

Some Definitions

The content of this section is long overdue. Through the book, we introduce numerous concepts that are peculiar to the identity management space. However, we have tried to avoid as

Decoupling: A Winning Pattern

The Identity Metasystem aims to decouple identity-related operations from their actual implementations. Solving a problem by adding a level of abstraction is a very common technique in computer science, and the examples of spectacular successes abound. The ease of use that we enjoy with computer networks today is perhaps the most visible instance of successful decoupling. In the late 1990s, every software developer who wanted to use any network capability was forced to target a specific protocol. A program written for working with Token Ring was different from one performing the same functions but designed to work on Ethernet. Software vendors needed to know which protocols were available on the customer LAN. Customers needed to know which protocols were supported by the products they bought. Any change had to be addressed by modifying the source code, with significant time and effort investments; and the contrasting requirements of different software packages drove network administrators crazy. Then something magical happened. The TCP/IP protocol started to enjoy widespread adoption on a growing number of platforms. TCP/IP made immaterial to developers the question of whether the target system supported Ethernet or Token Ring. You could program directly against TCP, and the actual protocol availability became a deployment problem. Today you don't have different browser executables for every conceivable protocol, and the same can be said for every network-enabled application. The ultimate proof of the soundness of the approach is the grace with which Wi-Fi, a protocol that was not even invented at the time of the creation of TCP/IP, was integrated into software systems. Again, the code that makes your browser tick is exactly the same whether your laptop is connected to an Ethernet cable or your traffic rides radio waves. This is the promise of agility and future-proof robustness that the Identity Metasystem brings to the world of authentication schemas.

much as possible the usage of many of the loaded terms that you find in the literature. It was not easy, but we hope that this model favors the unbiased understanding of the new ideas we are discussing.

It is now time to define more rigorously some of the terms we have been using loosely and substitute some of the words we used with specific identity-related terminology. We will not give all the definitions here, but we will keep introducing new concepts as appropriate throughout the rest of the book. For example, the definition of *subject* is delayed until the section "Roles in the Identity Metasystem," where the context is ideal to understand its function and importance.

Claim

A claim represents a fact about something or somebody. Better. A claim is a statement that a certain fact applies to something or somebody. As such, it is subject to verification. In other words, you can accept or reject the claim based on your beliefs, knowledge of the situation, and so on. Classical examples of claims about people are "Bob was born in 1956," "Bob is a Belgian citizen," "Bob belongs to the 'Managers' user group at Contoso.com," "Bob has green eyes," "Bob can buy $5000 worth of merchandise," "Bob is really the one who received this ticket from the TGS." Claims about things are no different: "Contoso's public key is Fx0Ex0…," "Contoso has its main office in Las Vegas," and so on.

Those claims may or may not actually apply to Bob or Contoso. The way in which you get to decide one way or the other accounts for a good part of the entire identity management process.

(Digital) Identity

As central as this concept may be to the topic of this book, the definition of (digital) identity is largely unimpressive:

A (digital) identity is a set of claims made by a subject about itself or another subject.

In the section "Roles in the Identity Metasystem" we will see what we mean exactly by *subject,* abbreviated S. For the time being, you can just substitute the occurrences of the word in the definition with the same "somebody or something" we used in the section "Claim."

Remember the discussion about the difference between credentials and digital identity in section "HTTPS, Authentication, and Digital Identity" from Chapter 1? The airline example should have provided an intuitive idea of what we mean by identity in this context. Now that we have a definition, we can refine the concept a bit further.

It is who asserts the digital identity that determines whether you will believe the claims in it

One thing you might notice from the definition is that an identity is made up of claims all coming from the same source; given the fact that claims may or may not actually apply, what you know about that source may influence what you believe to be true for the entire identity as opposed to considering claims one by one. There will be more (*much* more) about this later in the section.

Identities can be self-asserted

Another interesting thing you may notice from that definition is that an identity can be self-asserted. It is perfectly legitimate for somebody to make claims about himself. This actually happens all the time on today's Internet. When you sign up for an Internet service and you are asked to fill in a profile, you are making claims about yourself. Again, this is an important consideration and will be explored at length later in the book.

Trust

The concept of trust is pivotal in the IT security literature, and it can certainly elicit interesting philosophical digressions. In this context we will be much more prosaic, and we will simply define trust as *the willingness of a subject to believe the claims asserted by a certain other subject.* If Alice trusts Bob, any claim Bob will make will be considered true by Alice. There's that little matter of making sure that Bob is really who he says he is and verifying that the claims are actually coming from Bob; but after that is taken care of, Alice will believe just about anything. Technically, that is not strictly true because Alice's trust for Bob may be bounded only to certain areas. However, for the purpose of the explanations in this text, we can safely think in terms of unbounded trust.

Verisign says, via a certificate, that this website is "contoso.com"? Your browser is happy. Your government says, via a difficult to fake ID card, that you are over 21. Your bartender smiles and pours Chianti in your high-stem glass. That's trust.

Roles in the Identity Metasystem

The Identity Metasystem abstracts the entities and processes involved in identification operations.

The various actors participating in the transaction are perhaps the first things that need to be modeled, the basic blocks from which we can start to build our Metasystem. Understanding the invariant characteristics of relationships and mutual expectations is a key step toward successfully capturing the essence of the process. Observing the recurrence of such features across many different identity-related transactions leads to the definition of some archetypes, or *roles*, which successfully describe the behavior and the properties of all the actors involved. Substantially, if an entity participates in an identity-related process, you can always represent such an entity in the Identity Metasystem with one or more of those roles.

The Identity Metasystem distinguishes three possible roles: subject (S), relying party (RP), and identity provider (IP). As the following descriptions will clarify, those roles describe perfectly natural behaviors, in full agreement with the intuition; in fact, they are perfectly suitable for describing identity-related processes happening in the offline world, too. That should not surprise too much. We are rebuilding a system from the ground up, explicitly to get things right, free from the artifacts and aberrations derived from implementation details and historical burdens.

The next three sections introduce the three roles. In the section "The Dance of Identity" later in this chapter, we examine how those three roles contribute to propagate identity information.

Relying Parties

Relying parties consume identities

A relying party, often abbreviated RP, is an entity that consumes identities. An RP is typically something or somebody who provides a service that is intended to be enjoyed by a restricted audience. To make sure that the access is granted only to the rightful crowd, the RP requires receiving an identity from the requestor.

The wine seller in the example from the section "Minimal Disclosure for a Constrained Use" is an RP; so is any website that requires you to authenticate yourself before accessing its services. If you examine the section "The Babel," from Chapter 1, you will see that every authentication scheme described includes an entity that plays the role of the RP: intranet services requesting a certificate form a smartcard, HTTPS endpoints asking for a certificate via SSL authentication, the "service B" described in the "Kerberos" subsection. In SAML, the service requesting the caller identity is even called relying party!

The RP is a powerful invariant of identity-related systems. Its requirements are among the main reasons for which we need an identity system in the first place.

Subjects

Subjects have identities

We have already used the term *subject* a number of times throughout the book, relying on its common meaning. From a definition standpoint, a subject is just something or somebody who owns a digital identity. From the role definition point of view, however, it is worth considering the definition in more detail.

Anybody can be a subject

In the section "Directed Identity," we introduced the differentiation between omnidirectional and unidirectional identities. The former type of identity can often be assigned to every actor in a transaction, or at least to all the ones that exhibit one-to-many relationships. That basically means that the label "subject" can be applied to many entities in an identity system, and therefore its usefulness as a role-differentiating factor seems pretty unlikely. In the context of the Identity Metasystem roles, however, we usually intend the subject as one entity whose unidirectional identity comes into play. That does not mean that the entity cannot also own omnidirectional identities. Instead, it means that for purposes of modeling the behavior of an entity in the subject role in an identity transaction, we will consider only the unidirectional aspect. Translating the example in the section "Directed Identity" into Identity Metasystem terms would result in something like this: If the RP is the actor who consumes identities, the subject is the entity whom the consumed identity is about. If the wine seller plays the role of the RP, the buyer is the subject; it is the buyer's identity, in the sense of the claim defining his age, that the wine seller will want to verify ("consume").

Identity Providers

Identity providers assert identities of the ones they know about

The concept of IP is extremely natural. It models a role that is practically omnipresent in real-life situations in which people handle identities. Unfortunately, in traditional online authentication schemes, the IP is implicit or is an emergent property of the system, making it difficult to weave into the system the requirements associated with the role.

An identity provider, abbreviated IP, is an entity that issues digital identities. An IP is the entity that asserts the claims constituting a digital identity, typically in virtue of the relationship that associates it to the subject owning that identity. The list of examples from the offline world is endless. Governments can emit claims about their citizens; employers can issue claims about their employees; a department of motor vehicles can claim that a certain individual can lawfully drive particular vehicles; an airline can declare that a given individual is a passenger of a certain flight; a doctor can declare that a specified patient is fit for physical activity; a department store can award a customer with loyalty privileges. A very important example is the one in which an individual makes claims about himself, such as declaring his home address on a feedback form in a restaurant.

Identities issued by identity providers are effective to the extent that the IP is considered an authority in the current context

Note that in all the previous examples the IP was actually competent in terms of the kind of identity information mentioned. A government is a natural IP for its citizens because it actually owns the information involved (such as the passport number), and it has the appropriate means for managing it (such as demographic archives). Every entity aware of preceding facts will consider the government an *authority* in the matter of its citizens. In other words, it will *trust* the government (as *trust* was defined previously). This simple consideration gives us the last piece for fully translating the wine seller example in Identity Metasystem terms. The wine seller is the RP, the buyer is the subject and the government is the IP that provides the buyer with an identity (for example, in the form of a picture ID document). The RP trusts the IP and therefore accepts the claims on the document as true and acts accordingly, granting or denying the buyer request according to the rules.

IPs have always been there. The big shift is modeling them explicitly

Explicitly acknowledging the existence of the IP role is a powerful shift in perspective and helps to reconsider many aspects of identity-related transaction.

One of the concepts that surfaces more clearly thanks to the idea of IP is the identity context. Different RPs will grant their trust to diverse IPs, according to the service they offer or the relationship they themselves have with the IPs. In the offline world, you would never try to board a plane just by showing your driver license, nor would you attempt to get a discount at the local department store by waving your passport. Yet, as mentioned in the section "Consistent Experience Across Contexts," with today's online-authentication system, errors of that magnitude are not uncommon. Expressing identities as collections of claims was the first step toward clarifying the information flow: Explicitly stating the issuer of those claims, and its trust relationship with the RP requesting them, is the step that finally defines the transaction details and helps the subject to make informed decisions.

The concept of IP provides a useful model for describing scenarios in which the identity is self-asserted

Another important effect of introducing the concept of IP lies in the reinterpretation of transactions in which the identity information is claimed by the subject itself. In today's online world, many of the low-value services (typically the ones for which you are not charged) do not require the user to be endorsed by any specific IP. The authentication operation will just verify that the current requestor owns the credentials associated with a certain signup profile. That signup profile, created at registration time, is the subject identity. Some portion of the profile will have been entered by the subject itself, and hence it would be considered self-asserted. Name, surname, and email are typical examples of self-asserted claims. Some other portions of the profile (such as the last pages visited on that website in the former session) may contain information that belongs to the RP itself. The Identity Metasystem model allows the self-asserted portion of the user profile to be described as a full-fledged identity, issued by the subject to itself. In other words, the requestor simultaneously plays the role of the subject and the IP. Such an arrangement gives back control and awareness to the user, who can now maintain and disclose information at a finer level of granularity. Above all, however, the use of an IP in the case of self-issued claims provides a level of consistency that can finally satisfy the seventh law, "Consistent Experience Across Contexts." Windows CardSpace expresses self-issued claims via an artifact named Personal Card, which concretely realizes the advantages of the last scenario described here. Parts II and III of this book delve into the details.

The implications of the introduction of an explicit IP role in the system are profound and cannot all be covered here, but you will see more and more of them as the Identity Metasystem is described in further detail throughout this chapter.

In summary, an IP is the first occurrence of the word *subject* in the definition of digital identity (see the section "(Digital) Identity"). It is the entity that asserts claims about another subject, typically with regard to the relationship between the two. The digital identity is a currency that a subject can spend with a certain RP if the latter trusts the IP that minted it.

Freeing the "Hostage Identity"

In Chapter 1, in the section "HTTPS, Authentication, and Digital Identity," we encountered the concept of hostage identity. The identity of the user, intended as collection of claims, lives on the website itself, and it is "unlocked" by a successful user authentication. When this happens, the content of the claims can influence the behavior of that website but no others—if you disregard the few cases of business partnerships grouping together multiple entities (see the example in "User Control and Consent" and "Minimal Disclosure for a Constrained Use").

With the new model, all this can change. The Subject can obtain its identity from an IP, and the website (which clearly plays the role of an RP) does not need to keep those claims buffered anymore. The Subject can use the same collection of claims with any other RP that trusts the IP. The hostage is free. This is a true game changer, and it's natural to wonder how it can impact current practices. As this chapter unfolds, things will get clearer. Furthermore, Chapter 6, "Identity Consumers," is entirely devoted to IPs and explores those issues in depth. In this sidebar, we address an apparent contradiction induced by the introduction of the three roles. Now that an RP relies on an IP for releasing identities, aren't we outsourcing authentication? Didn't we say in "Justifiable Parties" that outsourcing authentication is bad?

The point is subtle but important. When an RP requires the S to present an identity obtained from an IP, it is asking S to present itself as a "customer" of the IP as opposed to a customer of the RP. If you are using an automatic kiosk for checking in for a flight, you can swipe the credit card that you used to buy the ticket. First and foremost, your ownership of the credit card proves that you are a customer of the credit card company; then, it is also a moniker for your record in the airline company back end. The airline didn't outsource its authentication operations to the credit card company. If you swipe your spouse's credit card, the system will not let you in. Furthermore, the data about the seats and whether the ticket allows access to the lounge is still on the airline's database, as opposed to the credit card company's. With IPs and RPs, it is almost the same. The RP trusts the fact that the S is recognized by the IP because it is able to present an identity from the IP. But that does not imply that RP will not perform any additional controls, nor that all the data relevant to the transaction must come from the IP. In fact, some data is pertinent only to the relationship between the S and the RP, and therefore they are not supposed to be "freed." Using terminology that we introduce later in this chapter (see the section "Identity Metasystem Components as WS-* Features"), you can say that the data should be kept in a user profile rather than a token. We revisit this topic at length in Chapter 6.

In summary, the model based on the idea of an IP is dramatically different from the outsourced authentication that the first Passport proposed. Although an RP relies on an IP to assert claims about which it is competent, in the previous example Passport would do the equivalent of storing the seat position and the luggage allowance on the credit card back end.

Components of the Identity Metasystem

The preceding section introduced the roles that an entity can possibly play in an identity-related transaction. You can verify identities (RP), you can have your identity verified (Subject), and you can provide an identity to somebody (IP). This is a beautiful model that also applies nicely to the offline world. However, we need to lower the abstraction level if we want to give a practical answer to the problem we decided to solve: adding an identity layer to the Internet.

Let's take one step back and gather our thoughts. What do we know so far? We want to solve the problem of propagating identities through the Internet. We said that we want a system of systems that would accommodate existing and future technologies in a single Metasystem (as opposed to yet another technology that would compete with the current and future offering). We have the laws, which warn us that the only constants on the Internet are diversity and change.

The "Microsoft Vision for an Identity Metasystem" white paper, the manifesto of the Identity Metasystem, coalesces the preceding consideration into a need for five key components, as follows:

- A way to represent identities using claims
- A means for IPs, RPs, and subjects to negotiate
- An encapsulating protocol to obtain claims and requirements
- A means to bridge technology and organizational boundaries using claims transformation
- A consistent user experience across multiple contexts, technologies, and operators

The list of components could be rearranged in different ways, but we chose to maintain the original criteria for the sake of coherence with the rest of the literature on the S. The following sections explain the components one by one, tying the definitions to the concepts introduced so far.

Claim-Based Identities

Identities are made of claims

At this point in the text, the reader is familiar with the concept of digital identity. In Chapter 1, we observed the shift from blind credentials to authentication in the section "Ascent"; in the section "HTTPS, Authentication, and Digital Identity," we gained an intuitive understanding of the concept of digital identity, where the frequent-flyer example showed a first instance of claims usage; in the section "The Babel," we observed how some technologies incorporate the idea of claim. In this chapter, we gave a formal definition of *claims* and *digital identity* in the section "Some Definitions." The reasons why an identity is well modeled by a set of claims have been given throughout the entire text. Now that we have defined the key roles and the relationships among them, it is natural to adopt claim-based identities as the currency exchanged in the Identity Metasystem.

Negotiation

The various participants in the Identity Metasystem support many different identification technologies. How can we achieve interoperability? One important component of the solution lies in the need for a negotiation protocol.

If two systems are capable of communicating in many different ways, they have to negotiate to discover which ones will work for both

Let's introduce what we mean by negotiation with an example. An Italian person and a Chinese person, perfect strangers, go to an international conference. They meet in the elevator. The Chinese person says to the Italian person "你好吗" and the Italian person answers "Non capisco!" As soon as it's clear that they can't understand each other, they shrug and part ways.

Imagine the same scene, but this time the two are wearing the conference badges mentioned in the section "Directed Identity" that identify the languages they speak. The badge of the Italian person says "Italiano, English"; the one of the Chinese person says "中文, English." This time the Chinese person will know that if he wants to be understood he can speak English. A glance at the two badges is enough to understand each other's capabilities and negotiate a common ground.

The Identity Metasystem provides a frame of reference through which entities can negotiate which underlying technology to use

The same principle can be applied to accommodating the diverse technological capabilities of the entities involved in an identity-related process. The Identity Metasystem should provide a means through which the various parties can negotiate which technologies among the ones supported will be used for that specific transaction. If a subject can express his identity with SAML or Extensible rights Markup Language (XrML), and the RP he's invoking can accept Kerberos or SAML tokens, the Identity Metasystem will provide a way for the two to agree on using SAML. One frequent question that arises at this point is what happens when there is no match. If the subject supports only X.509, and the RP supports only Kerberos, there's no way for the two to engage in a transaction, at least until one of the two acquires a capability compatible with one of the other party. The negotiation protocol cannot perform miracles and instantly make Italians speak Chinese; however, it is still useful for gaining knowledge of the requisites. It is important that the negotiation phase be embedded in the Metasystem, instead of being left as an explicit integration task to the parties, so that the format in which requirements are expressed is as formal as possible and the stage is completed without imposing burdens on the parties' implementers. In the section "WS-* Implementation of the Identity Metasystem," we describe WS-MetadataExchange, a concrete example of a negotiation protocol that enables querying web services for dynamically discovered policies.

Negotiation is a necessary step in a system of systems

Because the Metasystem does not define an authentication technology of its own, reaching an agreement on that requirement is a necessary condition for any transaction to take place. It is also important, however, to make sure that all parties understand other kinds of requirements less bonded to implementation details. In the wine seller example, the merchant needs to know the age of the subject. This is a requirement that the buyer needs to be aware of and understand if he is to decide whether he wants to disclose the requested information. The fact that the merchant will accept only claims from a government-issued ID is again information that needs to make its way from the RP to the subject. The set of requirements of an RP is said to be its policy. The IP has policies, too, as discussed later in the chapter.

Encapsulating Protocol

As the negotiation takes place, the information must actually flow according to the roles and the rules of the transaction. The subject needs *some way* to retrieve his identity from the IP, and the RP needs *some way* to receive it.

Every technology transmits data in its own way; a Metasystem needs to provide a generic encapsulation protocol

The existing technologies already have their own ways of representing identity and moving it from node to node. However, those methods will not interoperate, and therefore they need to be abstracted away. The Identity Metasystem needs to define a protocol that presents a common model to every participant so that no specific technology needs to be understood for establishing a connection; such a protocol, however, should also enable effective transfer of information according to the rules of the particular technologies. The latter is possible in a sustainable and future-proof fashion only if the Identity Metasystem is not required to understand the technicalities of every technology. It should be able to transfer that data without depending on features and peculiarities of the formats.

In the previous section "Negotiation," we saw an example in which two parties agreed to use SAML for their transaction. An encapsulating protocol allows the Identity Metasystem to put in practice that decision by transporting SAML information as it would have done for Kerberos or any other technologies (that is, without really knowing anything about how to interpret the SAML format).

Claim Transformers

In the examples provided so far, we have been pretty loose in our usage of claims. The wine merchant mentioned previously wanted to know the age of the buyer, but we didn't bother to provide more detail about the format in which that information should have been codified. We took for granted that the merchant could, with little effort, extract that information from a driver's license or from a foreign passport without much premeditation.

Well, we have reached one of the limits of the metaphor. Computer systems are much pickier than bartenders (or wine sellers), and the reasons and business models that require online identification are much more complex than our canonical example.

There will always be some differences in how entities use claims for modeling scenarios

Consider for a moment every home-banking application up and running on the Internet today. Nearly every one of those applications, and the corresponding back end, has a construct that represents the concept of an account number. The semantic of an account number is fairly unambiguous, even if some local shades of meaning are possible. Yet the representations will greatly vary from bank to bank. If you were to make those home-banking applications participate in the Identity Metasystem, their natural role would be an RP. The policy of those RPs may state that the subject's identity should contain an account number; however, because we are talking about computer systems, the way in each bank indicates an account number will make a difference. For the bartender, the DOB (Date of Birth) field on the driver's license is happily equivalent to the "Birth Date" field on the passport; for a computer system, `AccountNumber` is very different from `Account_Number`. This is a very easy example because banks and financial institutions already participate in standard definition bodies, and therefore they can come out with canonical claims representing the concepts inherent to their specific domain of knowledge. The point here is that an apparently minor difference can make or break the feasibility of a project when we talk in Internet scale, and the Identity Metasystem must be able to plan for and accommodate those differences. Those are just principles of good service-oriented architecture. Reducing the coupling between parties reduces unnecessary dependencies and leads to a more robust system. Before talking about how the Identity Metasystem copes with the incompatible claims problem, let's examine a slightly more complex example.

Crossing company boundaries is a scenario that often requires claim transformation

In the sections "User Control and Consent" and "Minimal Disclosure for a Constrained Use," we introduced an example in which a company is in partnership with a supplier, a hardware vendor. We mentioned that one of the claims that the subject should present to the hardware vendor RP is "spending limit." Who is the IP in that scenario? The natural choice is the employer. After all, purchases within that application happen in the context of the company-supplier partnership, and it is only natural that the latter will restrict the service to employees only. Hence, the employee's identity must be issued by the employer. The employer, however, might not actually know what the spending limit of the employee is. What if the value fluctuates following some business rules specific to that vendor? The agreements between the two parties may state that there's a monthly buffer, and beyond a certain threshold only managers are allowed to make expensive purchases. Sure, the employer may incorporate those business rules in its IT system; however, that would not scale at all because it would have to do so for every partnership it entertains and differentiate all expenses as they are made as opposed to keeping a single bucket sorted out at invoicing time. It is much easier, and far more natural, to leave that function to the supplier. The hardware vendor knows how much the employer spent so far because it has a good business reason for knowing it. It has yet to invoice it. It also knows the rule. A manager can spend even if the preordained buffer has been depleted, whereas nonmanagers will have variable allowance. In summary, the employer's IP can issue to the subject claims it is competent to emit, such as whether the subject belongs to the category Managers; the supplier's RP needs to know the spending limit of the subject, and the supplier knows how to derive that value just by knowing whether the subject is a manager. The solution is straightforward: We need a construct that performs claim transformations applying the business rule previously described.

Claim transformers can insulate architectures from changes and incompatibilities

Claim transformers are the ultimate decoupling devices. They can help reduce the technical and business differences between identity representations. They can handle naming issues, translating incoming claims corresponding to the same concept in a format understood by the RP; they can apply business rules by examining incoming claims and expressing the implications in terms relevant to the RP business; and they can resolve format incompatibilities, repackaging and transforming claims from one technology to another. Claim transformers are also the element that makes complex trust-chaining scenarios possible. A company that sells houses may only consider candidates who have been certified as eligible by a consulting firm. The consulting firm may trust the statements from a pool of banks for issuing eligibility certificates. The bank where you keep your main account may be part of that pool of banks. A claim transformer is the means through which the trust chain can percolate from you to the house seller. Your identity of bank customer can be sent to the consultancy firm, which in turn will issue an identity that satisfies the house seller.

Claim transformers are one vital component of the Identity Metasystem. There will be quite a few scenarios in which claim transformers will not be necessary. If all parties in a transaction understand the semantic of the claims required, they can all find a common technological ground, and there are only single-hop trust relationships, so the claims can be consumed without further processing. However, those scenarios cover only the simplest and cleanest situations. Even if in the future the semantic Web or a similar movement leads to a very large base of commonly accepted claims, there will always be scenarios in which the trust must be brokered, in which new technologies must be integrated, and in which some organizational gulf must be bridged.

Consistent User Experience

The importance of a consistent user experience cannot be stressed enough. In Chapter 1, in the section "The Babel," we invested some time to understand in depth how cryptography and current authentication protocols address the safety of identity information transfer; however, we also saw that the transfer is only one of the phases in which data is at risk. The section "Malware and Identity Theft" describes attacks in the information-entering phase, which are ignored by all the protocol schemes described so far. Now that we have had a chance to understand how HTTPS works, we can see how nothing in the common practices based on it addresses attacks such as phishing.

The analysis that brought about the formulation of the identity laws had many occasions to uncover problems derived from poor user experience, widespread inconsistencies, and nonexistent planning for integration of the human component. That's the reason why at least two laws, "User Control and Consent" and "Consistence Experience across Contexts," address the issue explicitly.

A successful universal identification mechanism cannot address just the needs of machines, regardless of how clever its metaprotocols may be. Because the Subject role will almost always be played by humans, the peculiarities and modus operandi of human beings deserve at least the same amount of attention we devoted to integrating the software components of the system. The lessons learned, as summarized by the laws, must make their way into any implementation of the Identity Metasystem.

The Dance of Identity

In this section, we describe in Identity Metasystem terms a couple of classical authentication scenarios. By seeing the various components and roles in action, you will gain a deeper understanding of functions and relationships.

Note that the two examples are just the most basic templates. With the three roles and the five components of the Identity

What About the Attacks in the Information-Storing Phase?

The section "Consistent User Experience" deals with two of the three kinds of attacks we covered in Chapter 1, in the section "Malware and Identity Theft." What about the third kind, the attacks in the information-storing and -processing phases? The Identity Metasystem model can help in this case, too, but it cannot give guarantees. If the RP requires the subject to supply certain information, the subject can decide whether he or she wants to disclose that data or withhold it. Ultimately, however, if that data is required for performing the service offered by the RP, the choice is between using the RP or giving up. Organizing the transaction according to the Identity Metasystem is the best way to conduct the process in the best possible way; but after the information is in the hands of the RP, its destiny is bound to what the RP will do with it. The law of directional identity will prevent certain kinds of abuses, but it cannot prevent the RP from storing data in an insecure location. Fortunately, the concept of claim-based identity enables new scenarios in which the problem is eliminated altogether. Because subjects can now move their identities in the form of claim collections (see the box "Freeing the 'Hostage Identity'"), RPs are not forced to store much information about its users. RPs may choose to store the absolute minimum for authenticating a returning user, relying on the subject to provide all the information in the form of claims every time it starts a session with the RP. Simply put, what is not there cannot be stolen. Not every RP will be willing to follow such an extreme route, and some businesses will need to store information about their users in the form of profiles (again, see the box "Freeing the 'Hostage Identity'" for an example). In any case, the approach does not need to be pushed to its limits to be effective: RPs can choose to avoid storing certain classes of personally identifiable information to reduce their liability in the case of security breaches in their stores.

In summary, the Identity Metasystem model offers powerful tools for mitigating the effect of attacks in the information-storing phase, too; however, use of those tools cannot be enforced, and effective countermeasures are ultimately left to the competency of the RP.

Metasystem, we now have at our disposal the intellectual tools for modeling any identity transaction of arbitrary complexity.

The Canonical Scenario

In the most classic scenario, we have one instance of every role represented. We have one subject, S, one relying party, RP, and an identity provider, IP. The situation is completely straightforward: S wants to use RP, which in turn requires its callers to present an identity issued from the IP to authorize access. This is, once again, a generalization of our wine seller example: S is the buyer, RP is the seller, and IP is whatever government institution issued an identification document to the buyer, and Claim1 or Claim2 (see Figure 2-1) is the age claim. In the rest of this section, we explain Figure 2-1, pointing out what part of the Identity Metasystem is involved as the transaction unfolds. Note that because we are still technology-agnostic at this point, we simplify the sequence a bit (especially in Steps 3 and 4).

1. S engages RP in a negotiation to acquire RP's policy and requirements. RP states that it will consider for authentication only the users presenting an identity issued by IP, in SAML1.1 format and containing Claim1 and Claim2.
2. S goes through the experience of mapping RP requirements with S's capabilities. Namely, S checks whether it 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.
3. Assuming that S does have a suitable relationship with IP, S negotiates with IP the details about how the IP wants to be called (for example, with which technology).
4. S uses the information acquired in the preceding step to request an identity from the IP. The encapsulation protocol tunnels the specific technology that the IP requires to be invoked.

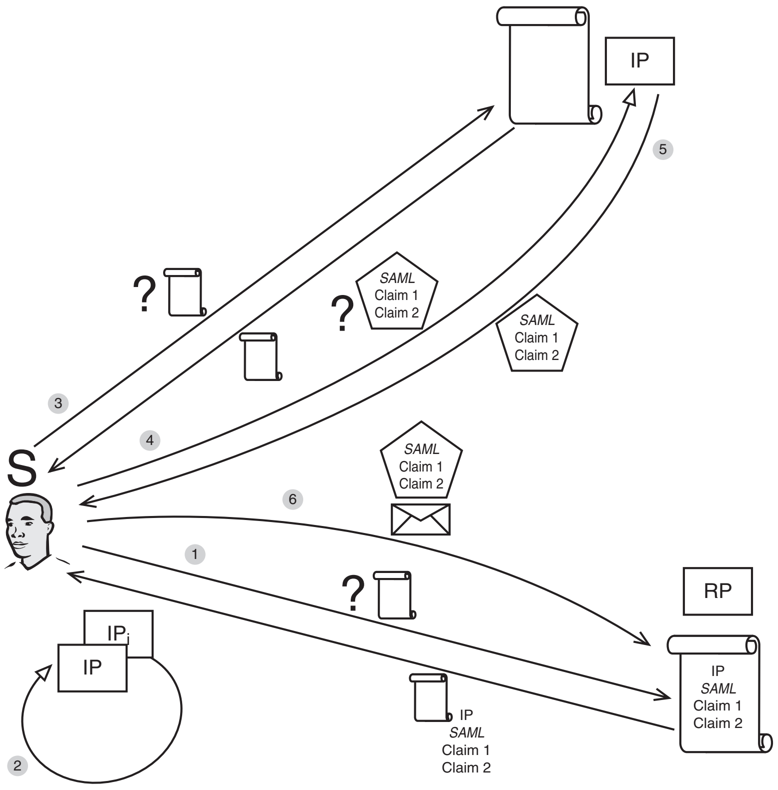


Figure 2-1 The diagram depicts the interaction among the three roles of the Identity Metasystem in the canonical scenario.

5. S receives the required identity from the IP. S examines 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 the RP.
6. If S decides to disclose, it uses the encapsulation protocol for transmitting the identity to the RP in accordance with the policy received in Step 1.

No technology prerequisites are imposed by the preceding sequence. All parties need to understand the Identity Metasystem; beyond that, however, everybody is free to use the technology of choice. Negotiation and encapsulation protocols provide the mechanism necessary to dynamically configure the system for automatic policy exchange and interoperability.

Brokered Trust

The brokered trust scenario generalizes the business partnership example developed in the section “Claim Transformers.” The situation depicted in Figure 2-2 includes four actors. A subject, S, a relying party, RP, and two identity providers, IP1 and IP2. Referring to the business relationship example mentioned previously, those elements map as follows: S is the employee that will make the purchase, RP is the web store of the hardware vendor, IP1 is the employer’s identity provider, and IP2 is the claim transformer, implemented in the form of an IP. A step-by-step description of the sequence follows.

1. S engages RP in a negotiation to acquire RP’s policy and requirements. RP states that it will consider for authentication only the users presenting an identity issued by IP2, in SAML1.1 format and containing the claim SpendingLimit.

Actually, My Driving License Is Still Valid

Steps 4 and 5 correspond to the request and issuance of a government ID document, respectively, in the offline-world example. In a real-life situation, you would likely already have a valid ID with you, and if it had expired, you would not be able to request and get a renewed one in the context of the wine purchase. However, in the online world, distance and bureaucracy mean nothing (or very little), so requesting that the IP issue a document on-the-fly is actually viable and guarantees freshness of the information. For more information about this misalignment between the example and the transaction please, see the section “WS-Trust” later in the chapter.

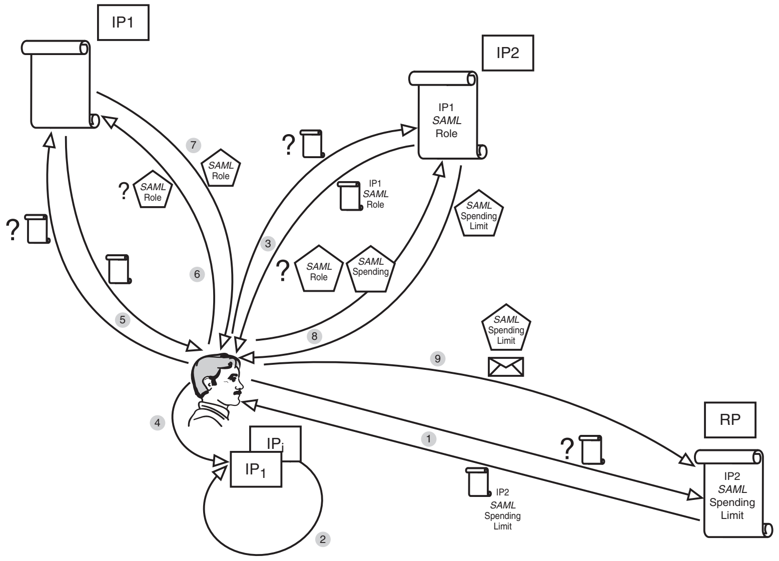


Figure 2-2 The schema shows the flow followed by a transaction in which trust is brokered through multiple IP.

2. S goes through the experience of mapping RP requirements with S capabilities. Namely, S checks whether it has a relationship with IP2 that would allow it to ask for a token of the right format and with the requested claims in it.
3. S does not have an existing relationship with IP2; hence, S engages IP2 in a negotiation, to acquire IP2's policy and requirements. IP2 states that it will consider for authentication only the users presenting an identity issued by IP1, in SAML1.0 format and containing the claim Role.
4. S goes through the experience of mapping IP2 requirements with S capabilities. Namely, S checks whether it has a relationship with IP1 that would allow it to ask for a token of the right format and with the requested claims in it.

5. S does have a suitable relationship with IP1. S negotiates with IP1 the details about how IP wants to be called (for example, with which technology).
6. S uses the information acquired in Step 5 to request an identity from IP1. The encapsulation protocol tunnels the specific technology with which IP1 must be invoked.
7. S receives the required identity from IP1. S examines the details of the identity, such as the content of Role, and decides whether it consents to the disclosure of that information to the RP and its trust chain.
8. If S decided to disclose, it uses the encapsulation protocol for transmitting to IP2 the identity it obtained from IP1. IP2 then issues to S an identity complying with the requirements of the RP.
9. S uses the encapsulation protocol for transmitting to the RP the identity obtained in Step 8.

It seems a long sequence, but it is really easier to do than to describe. The presence of the decoupling level provided by the Identity Metasystem enables the existing trust relationships to be leveraged automatically. A traditional identification technology would have required explicit out-of-band coordination, whereas the policy-based negotiation and the dynamic encapsulation protocol can self-organize a system that just works.

WS-* Web Services Specifications: The Reification of the Identity Metasystem

How do we create a real system that satisfies the requirements of the Identity Metasystem?

The Identity Metasystem looks very much like the solution we were searching for. However, what we have defined so far is still far from an implementation. We could devise systems in which negotiations and exchanges are made by throwing paper airplanes or swapping carrier pigeons and design those systems in a way that (given adequate bridging technology) satisfies the requirements we have described so far. From a more pragmatic point of view, giving the Internet an identity layer requires supplying a concrete, interoperable implementation of the components we encountered in the preceding section: 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; they must enjoy as wide a consensus as possible from the key players in the IT space and be accessible from the widest variety of platforms, contexts, and connectivity types. Sounds like a very challenging endeavor, borderline impossible.

Web services can get the job done

Didn't you hear this story before? Another technology in recent years exhibited a similar value proposition, trying to break free from platform or transport dependencies. That technology is loosely referred to by the name **web services**.

The pluralism dilemma was already dealt with by web services

Web services constituted a new way of exposing software to a distributed environment. The key idea behind web services is that, if the entire industry can agree on a set of common standards defining the key aspects of messaging, direct cross-platform interoperability is achievable. That vision gathered an unprecedented number of leaders in the IT industry. In an impressive collaborative effort, historical competitors set aside their differences and began a process of specification definition and standardization that is going on still today.

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. For this reason, the portion of the Identity Metasystem already in place today is largely based on web services.

In this section, we take a short break from identity and dedicate some time to understanding web services as a phenomenon and as a technology. We position them in the IT landscape and revisit basic principles, terminology, and the aspects more relevant to identity. After we cover the essentials, you will see how the various components of the Identity Metasystem are concretely implemented via advanced web services.

The WS-* Specifications

Developing distributed systems has always been one of the difficult problems of the IT industry. A piece of software that tries to communicate with another software entity across networks or other boundaries needs to address a number of problems. How do we route information to the destination? Which technique should be used for pumping data across the network? Which data format should be used? How do we ensure that the communication is secure and reliable? How do we guarantee that the communication happens in transactional fashion?

Before web services, interoperability had to be planned all the way down to painstakingly fine details

The standard way of dealing with those problems was represented by imposing on the designers the need to have complete knowledge of every node of the architecture. That meant knowing in detail which technology was used for developing all the software entities; which technology was used for exposing those entities on the network; and finally, all the painstaking details about the specific functions performed, the exact parameters exchanged, and their expected formats. The task was usually eased by using one single platform because doing so greatly reduced the number of variables coming into play. Component hosting systems such as COM+ and Java EJB emerged, and network middleware such as CORBA and again COM+ offered services for handling software communications. However, the results were often brittle. The tight coupling between software components (i.e., between who provided a function and who consumed it) made the systems extremely susceptible to changes and difficult to maintain. Cross-platform communication was also challenging and painstakingly achieved by ad hoc integration components and expensive bridges.

Software integration needs exposed the shortcomings of proprietary standards

As the IT world grew in importance and ubiquity, it became clear that those systems could not cope with the strain of increasingly diverse software environments coming from mergers and acquisitions, the need for integrating different software packages, and the dissolution of the boundaries between company information silos. Enterprise application integration knew a short period of popularity, but it was soon clear that there was the need for a strategic, long-term solution that would embrace different technologies. Again, you might have heard this story before!

The lesson provided by HTML was that of minimal requirements and resilience

While all those forces were building up pressure, part of the industry was trying to exploit the emerging ubiquity of the markup languages, such as HTML and XML, to devise a way to easily communicate across platform boundaries. Studying the universal success of HTML as the language of the Web, people realized that a large part of that success was due to its minimal requirements and resilience to errors and interpretations. As a result, in 1998 the first version of the Simple Object Access Protocol (SOAP) specification emerged. It was a very rough cut of what we know today; however, it defined the core of many key concepts still valid in the current vision. The specification defined how to project in-memory data types to XML format, a platform-neutral representation that can be understood without knowing anything about the technology that originated the data. It also defined a rough protocol for message exchange, again abstracting away the need to rely on a specific network transport protocol.

The initiative gained wide consensus among the main industry players and the analyst firms, with more and more important vendors joining the ranks of the specification proponents and backers as subsequent versions were released.

Interoperating is not enough in the enterprise world. You want security, transactions, reliability…

With the problem of sheer data transfer interoperability on its way to being solved, the market moved to consider the next stage: advanced communication capabilities. SOAP and its associated specifications (see the section "Basics" later in the chapter) didn't provide any way to secure messages from tampering in transit, nor was there a mechanism to provide confidentiality to communications; there were no means for a message sender to know whether a message actually reached its destination (and so on). Because the main purpose of web services was to connect loosely associated parties, shortcomings such as the absence of security were especially painful. There was still the chance of leveraging features of the actual transport—for example, sending SOAP messages through HTTPS would have guaranteed confidentiality; relying on that, however, would have partially eliminated the benefits of SOAP's platform-emancipation efforts.

Extending the SOAP specification with security features was a risky path. After having observed the issues with comprehensive but bloated solutions such as CORBA, SOAP was intentionally kept simple so as to keep its size and scope to a manageable level.

Modularity was the stratagem for keeping the web services specifications in a manageable format

The industry broke the impasse by creating additional web services specifications, each designed to solve a single aspect of the advanced communication problem. All those specifications built their new functionalities on top of SOAP and sometimes on top of other web services specifications. The idea was that implementers were not forced to deal with the full range of possible capabilities, but they could have chosen which specifications to implement according to the requirements of their systems. The specifications were all designed to work with each other gracefully, without imposing any unnecessary dependency.

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.

The WS- specifications are almost finished*

Today the WS-* specifications cover most of the key aspects of cross-platform software communication. Many of them already enjoy the status of industry standards ratified by entities such as OASIS or the W3C. Many products on the market, from the most diverse vendors or Open Source projects, leverage those standards for interoperating out of the box across different platforms.

Such ubiquity, coupled with advanced security capabilities, constitutes the ideal foundation for implementing a truly inclusive Identity Metasystem. In the next several sections, we familiarize you with some of the most important WS-* specifications, to the extent to which we can later understand what their roles are in the Identity Metasystem architecture.

Basics

The main idea behind web services is simple: Use a universally understandable format for describing your data and define a way to describe how you want to communicate with your software. That is achieved through a number of specifications that build on one another.

XML

The Extensible Markup Language (XML) 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.

To trivialize things a bit, XML is like a generic-purpose HTML. Where in HTML markups represent hints to the browser about how to render a page, in XML, markups just represent data. Figure 2-3 shows an example of an XML document.

XML is at the center of many important satellite specifications, such as XML schemas and XML namespaces, which we do not cover here.

SOAP

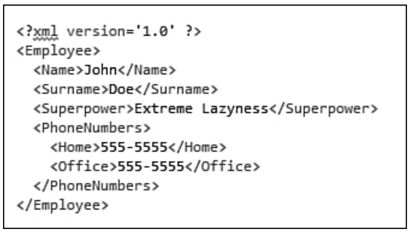
SOAP defines the message envelope format used for communicating with a web service

The SOAP specification, currently at version 1.2, defines what a web service message should look like and how it can be sent between two endpoints. SOAP represents everything using XML.

Imagine having a piece of software that performs the sum of two integer numbers A and B. A SOAP message requesting your software to perform the sum may look like that shown in Figure 2-4. It is an XML document that has a root element called envelope.

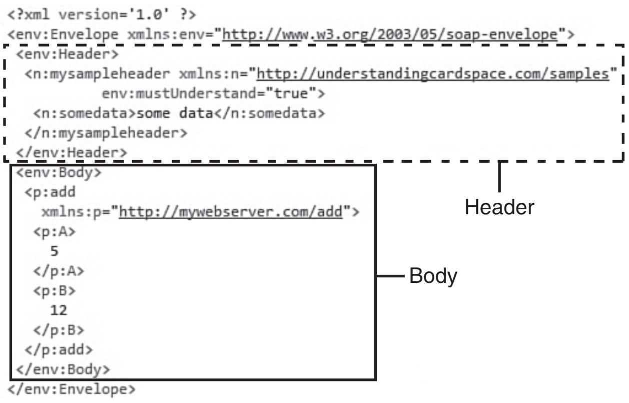
SOAP is extensible by design

The area between `<env:Body>` and `</env:Body>` is called the body of the message, and it contains the data for your software. The area between `<env:Header>` and `</env:Header>` is called the SOAP header element, and it's the key to SOAP extensibility. That is one area of the message where the infrastructure can weave additional information, enriching the communication with further capabilities. All the nonbasic WS-* specifications leverage this mechanism.



```
<?xml version='1.0' ?>
<Employee>
  <Name>John</Name>
  <Surname>Doe</Surname>
  <Superpower>Extreme Lazyness</Superpower>
  <PhoneNumbers>
    <Home>555-5555</Home>
    <Office>555-5555</Office>
  </PhoneNumbers>
</Employee>
```

Figure 2-3 A simple XML document



```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
 <env:Header>
  <n:mysampleheader xmlns:n="http://understandingcardspace.com/samples"
          env:mustUnderstand="true">
   <n:somedata>some data</n:somedata>
  </n:mysampleheader>
 </env:Header>
 <env:Body>
  <p:add
    xmlns:p="http://mywebserver.com/add">
   <p:A>
     5
   </p:A>
   <p:B>
     12
   </p:B>
  </p:add>
 </env:Body>
</env:Envelope>
```

Figure 2-4 A simple SOAP message

Simple Object Access Protocol? Not Anymore

At the time of its first formulation, SOAP was intended as a mean of accessing remote objects across platforms. In this, it was following the footprints of its predecessor, the XML-RPC specification. (RPC stands for Remote Procedure Call.) Hence, the original SOAP term was an acronym of the expression Simple Object Access Protocol.

As the idea of web service got further refined, it became clear that thinking of software in a remote location in terms of objects was not the best way of dealing with distributed systems. The idea of an object implies a certain degree of control from the caller, whereas in practice remote software is often entirely beyond the sphere of influence of its clients. Without going into too much detail here, web services moved away from the idea of exchanging objects as parameters and return values. Everything started to revolve around the concept of the message, intended as pure data without any logic associated with it. At the time of this writing, the latest SOAP version is 1.2. The current specification document explicitly drops the acronym. SOAP is about exchanging messages, not objects.

WSDL

WSDL describes the kind of messages that a web service accepts and produces

Once a caller knows that it can use SOAP for communicating, it needs to know which kind of data the software can handle. The Web Services Description Language (WSDL) absolves that function. Documents written in WSDL describe the operations exposed by a web service and the message format required (and returned) for every function provided. The idea of an explicit contract is very important in software development and in web services and service orientation is possibly even more important. However, further discussion of such is beyond the scope of this book. For a comprehensive coverage of the topic, consult *Understanding Web Services* (Newcomer, 2002).

WS-Addressing

WS-Addressing provides a rich way of referring to a web service

In Chapter 1, in the section "HTTP," we saw how web page addresses work. The address `http://www.bob.com/bob/homepage.htm` tells the browser to use the HTTP protocol for retrieving the HTML document `homepage.htm` residing at the path `/bob/` on the web server `www.bob.com`. The address is both a unique identifier and a way to retrieve the page. 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.

WS-Policy

WS-Policy advertises what is necessary for calling a web service while satisfying its requirement

WSDL describes the operations offered and the message formats required by a web service, but it does not give further details about any other requirements associated with the web service invocation. For example, a web service implementing a wire transfer may be invoked with the correct message format, but the software will not execute the operation unless the caller identifies itself using a certain authentication technique. WS-Policy provides a generic purpose for describing such requirements, which are said to be the policy of the web service. WS-Policy (and its sister specification, WS-PolicyAttachment) does not define any domain-specific policy assertion such as the one about authentication in the preceding sample. It is a generic mechanism for associating requirements ("policy assertions") to a web service, and as such it does not mandate any particular format. Other specifications, such as WS-SecurityPolicy described later, leverage this general-purpose mechanism for codifying requirements of a specific domain.

WS-Security

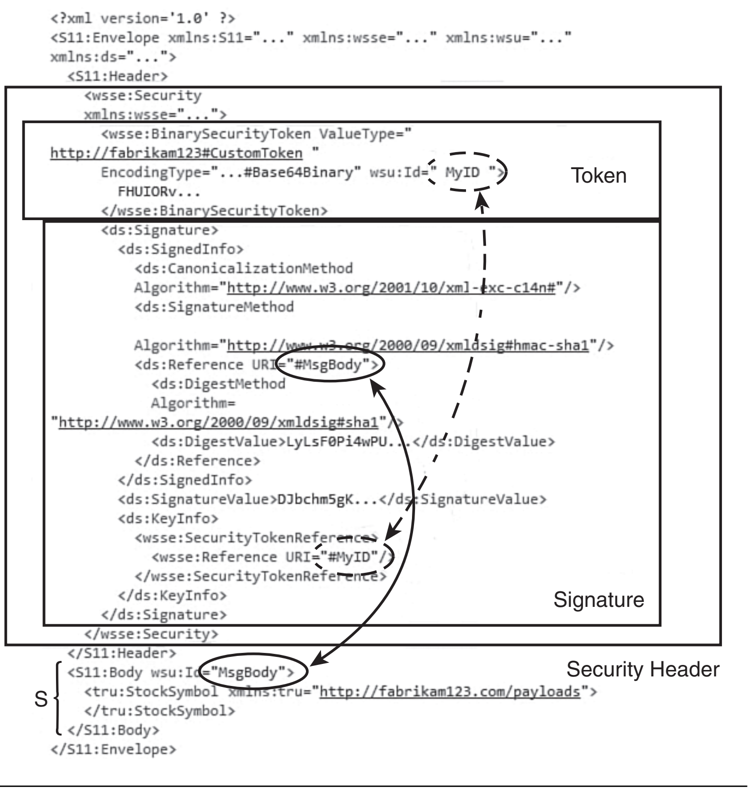
WS-Security was the first specification building on the extensibility capabilities of SOAP. Although the specification itself and its derivatives are fairly complex, the purpose of WS-Security is straightforward. It defines ways of protecting SOAP message exchanges and provides a means of transporting security-related information.

Security was the first advanced capability added on top of SOAP, and it leveraged the work already done for XML

Given the enormous success of XML, the industry soon felt the need to provide some security mechanism that could guarantee confidentiality and integrity to the new format, without giving up its cross-platform reach. As a result, the W3C devised two standards, XML Signature and XML Encryption, which describe ways of applying cryptography to XML documents (see the section "Cryptography: A Minimal Introduction" in Chapter 1). Such standards described extremely flexible operations, in which different parts of the document could be encrypted or signed using different algorithms or even different keys. WS-Security describes how to apply XML Signature and XML Encryption to a special kind of XML document, the SOAP message. Without going into the fine details, the peculiar structure of a SOAP message offers a natural way to apply the model. The message can be modified according to the intended operation—for example, by substituting the body with encrypted data—

while the SOAP header can carry a description of the cryptographic transformation that took place. The receiving end of such a message analyzes the content of a special WS-Security SOAP header, discovering that the body was encrypted using a certain algorithm and a certain key; if the receiver owns the corresponding key, he can now reverse the process and decrypt the body. The signature case is analogous.

Figure 2-5 illustrates such a SOAP message.



```
<?xml version='1.0' ?>
<S11:Envelope xmlns:S11="..." xmlns:wsse="..." xmlns:wsu="..."
xmlns:ds="...">
  <S11:Header>
    <wsse:Security
    xmlns:wsse="...">
      <wsse:BinarySecurityToken ValueType="
http://fabrikam123#CustomToken "
      EncodingType="...#Base64Binary" wsu:Id=" MyID ">
        FHUIORv...
      </wsse:BinarySecurityToken>
      <ds:Signature>
        <ds:SignedInfo>
          <ds:CanonicalizationMethod
          Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
          <ds:SignatureMethod

          Algorithm="http://www.w3.org/2000/09/xmldsig#hmac-sha1"/>
          <ds:Reference URI="#MsgBody">
            <ds:DigestMethod
            Algorithm=
"http://www.w3.org/2000/09/xmldsig#sha1"/>
            <ds:DigestValue>LyLsF0Pi4wPU...</ds:DigestValue>
          </ds:Reference>
        </ds:SignedInfo>
        <ds:SignatureValue>DJbchm5gK...</ds:SignatureValue>
        <ds:KeyInfo>
          <wsse:SecurityTokenReference>
            <wsse:Reference URI="#MyID"/>
          </wsse:SecurityTokenReference>
        </ds:KeyInfo>
      </ds:Signature>
    </wsse:Security>
  </S11:Header>
  <S11:Body wsu:Id="MsgBody">
    <tru:StockSymbol xmlns:tru="http://fabrikam123.com/payloads">
    </tru:StockSymbol>
  </S11:Body>
</S11:Envelope>
```

Figure 2-5 A SOAP message whose body has been signed via WS-Security. The header section contains the WS-Security header, which in turn contains a security token and the signature element itself. The solid line and arrowheads highlight the reference to the part of the envelope that has been signed (in this case, the entire body); the dotted line and arrowheads show the parts that associate the signature with the token containing the associated key. The section S, indicated by the curly bracket, shows the portion of the message that has been signed.

WS-Security does not introduce any new cryptographic algorithm, nor does it define any new source of keys. SOAP is a trade language, designed for bridging different platforms and technologies. To effectively support the SOAP mission, WS-Security needs to be able to accommodate existing security technologies and promote interoperability among those. If it sounds quite similar to what we have seen for the Identity Metasystem and existing authentication technologies, that's because it is.

WS-Security provides existing cryptographic technologies with a framework for securing elements of SOAP messages

WS-Security needs to be able to encrypt and sign SOAP messages by using the technologies available to its users: X.509, Kerberos, SAML, username and passwords, plus every present and future source of cryptographic material are candidates. However, there cannot be any explicit dependency in the specification; otherwise, the good cross-technology properties would be lost. The WS-Security authors devised a clever trick for solving the impasse. The specification assumes that the cryptographic material travels in a WS-Security security token, a generic data construct, and refers to it for defining the various encryption and signing operations on the message. Then they separately provided satellite specifications, named **token profiles**. Each token profile describes how to derive a WS-Security token from an existing security technology, mapping the peculiarities of the particular system to generic WS-Security features. For example, a web service may mandate that the body of the message be signed without specifying the kind of key used. A certain caller may sign using a security token derived from an X.509 certificate, whereas another may use a security token derived from a Kerberos ticket. As long as the receiving end can verify the validity of the signature—for example, by choosing by a pool of well-known certificates or by being part of the same Kerberos domain—everything goes as expected. If at some point in the future a new authentication technology is released, and a suitable token profile is defined, the new technology can be seamlessly integrated into the system without requiring any major change. This arrangement effectively decouples the security

The strength of WS-Security is in the generality of the idea of a security token capabilities of the protocol from the technologies actually available, allowing users of different technologies to speak a common tongue while still having a return on their investment on the platform of choice. Those are exactly the good properties we indicated as key requirements for the Identity Metasystem, at a lower abstraction level. As discussed in the following sections, the WS-Security token occupies a pivotal role in realizing an architecture coherent with the vision of the Identity Metasystem described thus far.

WS-Security Tokens and Token Profiles

At the time of this writing, the current version of WS-Security is 1.1. It is a standard ratified by OASIS. OASIS lists the following standard five different token profiles:

- Username token profile
- SAML token profile
- X.509 token profile
- Kerberos token profile
- Rights Expression Language token profile

Being part of the WS-Security standard, those token types can be safely used in scenarios requiring out-of-the-box interoperability; the profiles take care of describing the expected behavior in fine detail, such as using AssertionID or ID for referencing SAML assertions crafted using different versions of the SAML standard.

Nothing prevents vendors and customers from creating their own token profiles, to leverage existing investments in technologies not covered by the five profiles in the specification. As long as every actor who needs to use the new kind of token understands it, everything will work as expected.

WS-Trust

The section "The Babel" in Chapter 1 subdivided the authentication schemes into two big families: the ones based on certificates and the ones based on issued tokens. WS-Security can handle security tokens derived from both schemes, as long as the requirements expressed by the relevant token profile are applied. Every authentication technology based on issued tokens describes in its own way how a client can obtain a token. The two examples we have seen, Kerberos and SAML, perform that operation in very different ways. WS-Trust generalizes the token-issuance operation to WS-Security tokens. In other words, WS-Trust extends WS-Security with methods for issuing, renewing, and validating security tokens in a platform-agnostic manner. The advantage is evident. Whereas WS-Security assumes that you managed to create your token outside of your web service architecture, using some unspecified security technology, WS-Trust allows you to also model, in technology-agnostic fashion, the operations necessary to obtain tokens. Thanks to WS-Trust, web services–based systems can now enjoy the flexibility of issued token–based technologies with the added bonus of not being tied to any specific stack.

WS-Trust extends WS-Security with methods for issuing, renewing, and validating security tokens in a platform-agnostic manner

How does that all work? With its 75 pages of dense security considerations, the WS-Trust 1.3 OASIS Standard specification is a fairly complex document. A comprehensive description of the standard is beyond the scope of this book. However, it is of paramount importance to understand very well the main scenario and the associated terminology because it is the cornerstone of today's Identity Metasystem implementation.

WS-Trust introduces a special kind of web service, called Security Token Service (STS). To put it simply, the job of an STS is "transforming" WS-Security tokens. One token enters; another token exits.

An STS is a special web service that can issue security tokens

Let's assume that a certain client C wants to invoke a certain web service S. Let's also assume that S specifies in its policies that for security reasons it will accept requests only if secured by a certain WS-Security token, say a SAML-based WS-Security token containing a certain claim about C. C can ask an STS to issue the SAML token it needs for calling S. The request is performed by sending a special kind of message, whose format is described in WS-Trust, called a Request for Security Token (RST). The RST contains, among other things, the description of the kind of token that C is asking the STS to issue. The STS, however, will not issue tokens to just anybody. Because the SAML token required must contain a claim about C, the STS must make sure that is actually C who is requesting the issuance (read, the RST message is actually coming from C). Hence, C must secure the RST message in a way that will convince STS of the correct provenance of the message. In WS-Security terms, this means that C must secure the RST using some security token. For example, C may secure its request to the STS using a Kerberos-derived security token.

When the STS receives the RST, it examines the incoming token and checks that it was properly secured. If everything is as expected, the STS considers C's identity as verified and proceeds to generate the requested SAML token. The newly generated token is sent back to C inside another WS-Trust-defined message called a Request for Security Token Response (RSTR). When C receives the RSTR, it can extract the requested SAML token and use it for invoking S. Upon receipt of the invocation, S can extract the SAML token and verify whether the claim about C satisfies its requirements. Figure 2-6 shows the exchange just described. The end result is straightforward. The STS received a Kerberos token and issued a SAML token in return.

At this point, you might be asking yourself where the "trust" aspect in all of this is. The answer to that is simple: The scenario just described was just a fairly elaborate dance for allowing C to benefit from the trust relationship between S and STS.

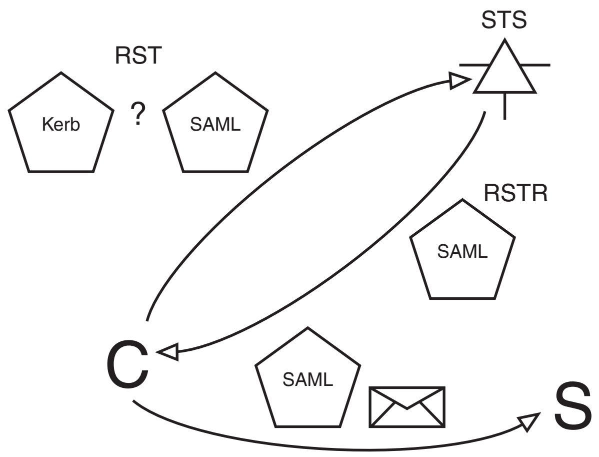


Figure 2-6 WS-Trust in action. To invoke S, C obtains a SAML token from an STS.

In the generic case, the reason for which C has to go to the STS goes beyond the sheer need of changing token format. Usually S does not trust C, so C needs to be endorsed by somebody who S trusts. Remember our ever-present wine seller example. In this case, the web service S is the wine seller, and the client C is the buyer. The claim requested by S is the age, and the picture ID that the buyer shows to the wine seller is the security token. Just as the wine seller trusts the age written on the picture ID because it is government issued, S trusts the content of the claim in the SAML token because the latter is coming from the STS. The analogy is not a perfect match. If it were, in the offline world it would mean that your driver's license (or any other ID document) would always be expired, and you'd have to get one freshly issued every time you need to show it to somebody. In that case, you would need to contact your department of motor vehicles on-the-fly, and they would want to verify your identity (maybe checking your passport) before issuing you a new license. As you probably have already discerned, the department of motor vehicles plays the role of the STS, and your passport has the same function as the Kerberos token in our diagram. It is okay that the analogy is not 100 percent accurate. Tokens and picture IDs have many similarities, but the former can be used in many more ways and enables scenarios that do not have a counterpart in the offline world. Besides, we dare the bureaucracy of any administration to issue IDs as fast as an STS can issue tokens! That said, there are still some instructional aspects of the analogy that would be useful to spell out. The wine seller knows that the picture ID shown by the client is true because it recognizes the government manufacturing (e.g., holographic serigraphy or special paper) and implicitly assumes that it is extremely difficult to forge. How can S be sure that the SAML token presented by C was actually issued by the STS that S trusts? The system is much more secure than the offline counterpart. The STS signs with its private key all the tokens it issues, so anybody knowing the STS public key can verify their source. Furthermore, the wine merchant compares the facial features of the client in front of him with the picture in the ID document, thus verifying that the document was actually issued to the buyer. In the web service world, C demonstrates that the SAML token was actually issued to it by being able to use the token for securing its request to S. In doing to, C is showing off that it knows a certain key that could have been acquired only from the RSTR that contained the token. There is no need to understand the details of that exchange. The bottom line is that S has cryptographic proof that C is the legitimate holder of the token, so the token cannot be fraudulently repurposed by others.

In summary, WS-Trust defines entities and messages for issuing WS-Security tokens via web services. The preceding example explored the scenario in which a client requests that an STS issue a token. However, the specification covers many other cases, such as issuance requests coming from services and token management beyond pure issuance (token renewal and validation being two examples). We concentrated on that scenario because, as we observed, it exhibits striking similarities with identity-related transactions we encountered elsewhere in the text. In the section "WS-* Implementation of the Identity Metasystem," we further clarify the parallel. The capability of WS-Trust of expressing trust relationships between parties will play a key role in the realization of an identity layer for the Internet.

SAML: Token or Protocol?

You might have noticed that throughout the text the term *SAML* appears very, very often.

As you read in the section "SAML" in Chapter 1, the SAML specification defines a protocol on its own. It has its own ways of dealing with token issuance, for example; and it tries to solve problems such as the single sign-on, which live at a different level of abstraction than the sheer WS-Security specification. How does that play with all the "technology-agnostic" rhetoric we used in the sections "WS-Security" and "WS-Trust"? The answer to that question is very simple. Apart from the section "SAML" in Chapter 1, every time we mention SAML throughout this book, we are not referring to the SAML specification in itself, but to the SAML token profile mentioned in the sidebar "WS-Security Tokens and Token Profiles." The SAML token format is extremely flexible and proved to be an ideal vessel for security-related information in many scenarios. Used in conjunction with the WS-Security token mechanism and the rest of the WS-* family of specifications, it lends its expressive power without introducing dependencies on any particular technology. Therefore, for the purpose of understanding the concepts presented in this book, you can safely ignore the protocol portion of SAML.

Also note that WS-Federation, briefly described in the section with the same name later in this chapter, presents a certain degree of overlap with the functionalities offered by the SAML specification; however WS-Federation is fully integrated in WS-* and works well with any part of it.

WS-MetadataExchange

Use WS-MetadataExchange for asking a web service about its metadata

WSDL and WS-Policy provide means to describe the web service to the world, or better, they help define the ways in which external callers are supposed to interact with the service itself. In the first years of web service existence, those documents were acquired by potential callers out of band or leveraging features of the specific web service stack implementation. For example, the Microsoft stack made the WSDL of a service available at one special address, obtained by attaching the string "?WSDL" to the address of the service itself. As the web service–based transactions grew in complexity, it became clear that there was the need to define how to acquire service metadata in a standard and programmatic fashion. WS-MetadataExchange is a protocol that fulfills exactly that purpose. It allows one caller to query one web service and obtain its metadata information, typically WSDL/policies.

WS-SecurityPolicy

WS-SecurityPolicy is a dialect of WS-Policy that deals with security concepts

WS-SecurityPolicy defines an assertion framework (that is, a collection of assertions and assertion operators) aimed at expressing security requirements for the invocation of web services. It 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. Although WS-Policy is generic enough to express any policy, it is good to have, for security, a set of standard assertions with a well-known semantic to which every platform and product can refer without further negotiations.

WS-Federation

WS-Federation builds on top of WS-Trust and WS-Security for modeling message exchanges in federated scenarios

We already encountered the concept of federation. However, it is worth revisiting the concept. A federation is a set of two or more entities, where resources of one entity can be accessed by identities belonging to another entity. If that sounds confusing, just think of the example offered in the sections "User Control and Consent," "Minimal Disclosure for a Constrained Use," and

"Claim Transformers." In that instance, the two entities were a company and its hardware supplier. The hardware supplier was offering access to its web store (the "resource") to the employees of the first company. The two formed a federation.

WS-Federation builds on top of WS-Trust and WS-Security, organizing the primitives offered by those specifications in a higher-level language suitable for modeling systems such as the example just mentioned. In practical terms, given a certain topology of clients, services, and STSs, WS-Federation establishes the sequences of messages that must be exchanged among the various parties for obtaining a certain result. In our simple example, the result is an employee of the first company accessing the web store offered by the hardware vendor, but WS-Federation is expressive enough to solve much more complex scenarios such as multicompany single sign-on. WS-Federation describes how to deal with those scenarios in synergy with other WS-* specifications. The case in which the actors are web services is described as the **active requestor** case. A requestor is active because, being web service capable, you can expect it to be able to use cryptography on the messages emitted and show off the ownership of keys.

A comprehensive solution, however, cannot ignore that many transactions are driven through the use of a web browser. A web browser cannot apply cryptography to messages in the same way as a web service can. Hence, this situation must be accommodated by opportunely devising message exchanges protected by transport security. WS-Federation devotes a comprehensive portion of its text for addressing the web browser case, which is referred to as the **passive requestor** case.

WS-Federation is a specification of key importance. The explanation we gave here does not even begin to scratch its surface. It is advisable to everybody interested in enterprise identity management to become intimately familiar with this specification.

The Identity Metasystem and the practices it enables are often defined as "user-centered federation." Whereas WS-Federation relies on automatic sequences driven by metadata and by inter-company partnerships, the Identity Metasystem can leverage the newfound user control for driving decisions with much looser relationships between entities. The two models are complementary, and they have ample areas of collaboration and synergy. Chapter 4, "CardSpace Implementation," discusses how Windows CardSpace handles federation in more detail.

WS-* Implementation of the Identity Metasystem

In the previous section "The WS-* Specifications," we devoted some time to better understanding the phenomenon of web services. Web services emerged in independence from the identity-related considerations we presented in this chapter, but they are the best tool at the industry's disposal for putting into practice the requirements discovered while formulating the seven laws and envisioning the Identity Metasystem.

Identity Metasystem Components as WS- Features*

Let's put the idea to test. Imagine that the three roles defined by the Identity Metasystem (subject, relying party, and identity provider) are implemented as web services. To be exact, we should say that every role will communicate with the other entities via web services. Holding on to that assumption, let's recall what the components of the Identity Metasystem were, as follows:

- A way to represent identities using claims
- A means for IPs, RPs, and Ss to negotiate
- An encapsulating protocol to obtain claims and requirements
- A means to bridge technology and organizational boundaries using claims transformation
- A consistent user experience across multiple contexts, technologies, and operators

The component-consistent user experience across contexts cannot be addressed directly by a protocol (even if it is the existence of a common metaprotocol that makes consistency possible to begin with). Therefore, we defer consideration about it until after the discussion on WS-*. All the other components find perfect fits in the entities and capabilities provided by the WS-* specifications.

A Way to Represent Identities Using Claims

The WS-Security token is the perfect fit for representing an identity

The obvious candidate for representing an identity in data exchanges is the WS-Security token. A token is self-contained and claim-based by design, so it owns the necessary expressive power for describing a digital identity as we defined it. The definition of token in WS-Security and the token-profiles mechanism avoids dependencies from existing and future authentication technologies, maintaining the potential to embrace them all. Finally, a token issued by an STS can be tracked with cryptographic certainty to its source. That makes the RST-RSTR transaction described in the section "WS-Trust" the perfect implementation of the process, followed by the S for acquiring an identity from the IP.

A Means for Identity Providers, Relying Parties, and Subjects to Negotiate

WS-Policy and WS-MetadataExchange provide an effective way of expressing and negotiating requirements

Web services architectures try to keep out of band communication to a minimum, aiming to expose all the information relevant to invocation via standard means. WSDL and WS-Policy, with its specializations such as WS-SecurityPolicy, make explicit to everyone the requirements that must be satisfied for being able to use a certain web service. The requirements can cover the most diverse areas, and they can certainly address things especially relevant to the metasystem such as expressing which authentication technology should be used. WS-MetadataExchange makes it possible to acquire such requirements directly online, keeping the need for coupling between parties as low as possible. RPs can easily use the tools above for expressing what it takes for engaging in business with them.

WS-Policy and WS-Metadata exchange can easily tell the subject that the web service of an online wine merchant requires a SAML token from the STS of the department of motor vehicles (driver's licenses), and that such a token must contain a claim with the age of the S. An S can acquire the relevant policies via WS-MetadataExchange and make a match between requirements and capabilities. An IP that would expose its identity-issuing capabilities by mean of an STS could specify its requirements using exactly the same specifications.

An Encapsulating Protocol to Obtain Claims and Requirements

WS-Security token generality makes WS-Security ideal as an encapsulating protocol

Because we implemented digital identities using security tokens, it follows pretty naturally that the encapsulating protocol is WS-Security itself. WS-Security defines how to attach and use security tokens to messages. Such a definition does not change regardless of the source from which the WS-Security token was derived, being it SAML, X.509, Kerberos, or any other technology. WS-Security serves the purpose of the encapsulating protocol very well.

A Means to Bridge Technology and Organizational Boundaries Using Claims Transformation

STSs are natural claim transformers

Claims transformation can be easily performed by an STS. Security tokens are flexible enough to provide the technology and claim types transformations for bridging differences in requirements such as the ones described in the section "Claim Transformers."

The Dance of Identity—Implemented by WS-*

Now that we have defined a mapping between the Identity Metasystem and web services elements, we can give concrete indications about how the sequences presented in the section "The Dance of Identity" can be implemented with technologies available today. We will revisit the two sequences, specifying how every step is realized with WS-*.

The Canonical Scenario

Again we have one subject, S, one relying party, RP, and an identity provider, IP. The RP is implemented as a web service; the IP offers its functions by mean of an STS. (An STS that offers IP functions is often referred to as an IP-STS.) S was and remains a person, a human user of the system. For the purpose of this walkthrough, we will assume that S uses some sort of agent that hides the complexities of the web services interactions. We will get back to that agent in greater detail in the section "Presenting Windows CardSpace."

With this web services mapping in mind, let's revisit the original sequence under a new light (see Figure 2-7):

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 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.
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).
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 WS-MetadataExchange for acquiring IP's invocation policy.
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.

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

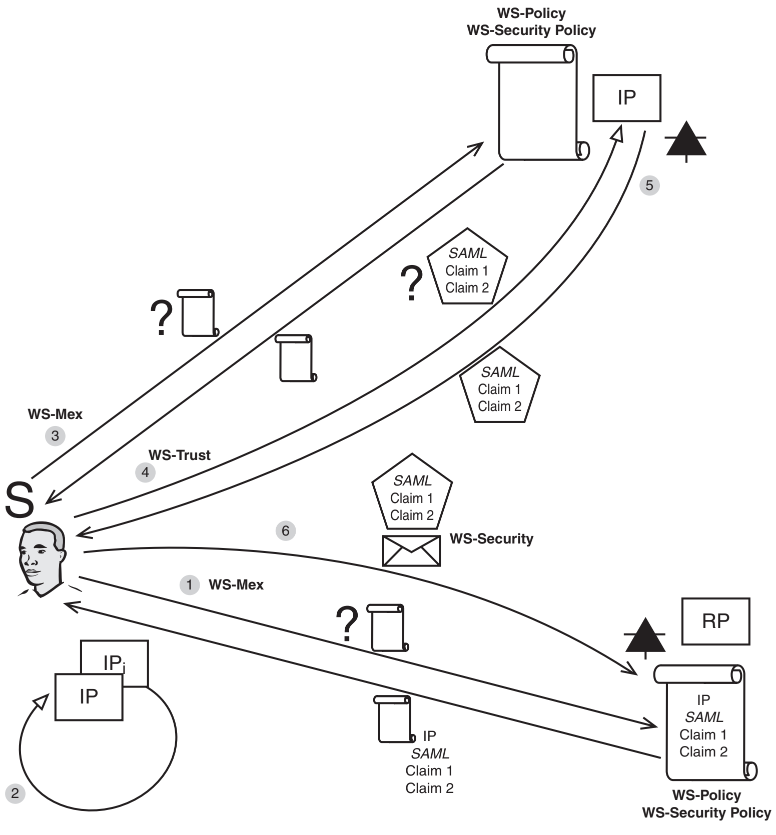


Figure 2-7 The schema of the canonical identity transaction, showing which WS-* standards are used for implementing every step

The preceding sequence uses only technologies in wide availability already today, yet all the requirements imposed by the Identity Metasystem are preserved. If all parties understand WS-*, a requirement that does not mandate any particular platform per se, the negotiation capabilities of WS-Policy and WS-MetadataExchange guarantee that if there is a match among the parties, it will be found. WS-Security ensures that the specific technologies are properly tunneled while maintaining a common abstract protocol, whereas WS-Trust guarantees that if there is a trust path between parties, the system will be able to exploit it for flowing identity information.

WS- is the only requirement here. Every entity can be implemented on any platform or technology*

Brokered Trust

The case of brokered trust is analogous to the one described in the section "The Dance of Identity." The rules of mapping to WS-* elements are the same ones demonstrated in the previous section. There is one thing that is worth highlighting: IP2 is still implemented as an STS; however, in the brokered trust scenario it performs pure claim transformation rather than sheer identity provisioning. An STS that performs that kind of function is called a **Resource STS**, or R-STS, because it takes care of mapping claims for a resource as opposed to providing identities for generic utilization. RSTSs are discussed more in depth in Chapter 4, in the section about federation, and in Chapter 6.

Presenting Windows CardSpace

At first glance, many of the Identity Metasystem requirements sounded almost utopist. Lucky for us, the WS-* specifications committees already covered many of the issues we had to face, including the toughest ones involving wide industry consensus, and now the Identity Metasystem can benefit from their work. What sheer protocols can't address, however, is the human integration aspect.

What About the Web Browser?

We have seen in detail how web services provide all the necessary power for implementing secure identity transactions. It is common knowledge, however, that as of today the vast majority of interactions on the Internet goes through a web browser. As observed in the section "WS-Federation," the web browser is passive. If S were to use a browser for performing Step 6 from Figure 2-7, and RP were a website as opposed to a web service, there would be no way of using WS-Security for applying the token to the invocation.

The case is easily addressed by using the same trick employed by WS-Federation (that is, using transport-based security in the segments of the schema that are not WS-* capable). Note that all the WS-Trust calls do not necessarily have to go through the browser; in fact, in the sequence in "The Canonical Scenario," those operations go through the S agent, which may be WS-* capable even if the main transaction is being handled by a browser.

WS- is great, but what about human integration?*

The hard-learned lessons from poorly usable systems are captured by the "User Control and Consent" law and, above all, by the "Human Integration" law. Expecting the user to understand WS-MetadataExchange and WS-Trust is possibly even more naïve than expecting the user to be able to assess the identity of a website from its SSL certificate. Having a solid layer of common protocols is a prerequisite for having a consistent experience across contexts. However, the experience must be good to begin with. Here, *good* stands for all the criteria established by the laws. The user must understand what is going on, he must be aware of his options, he must be able to make decisions in a natural fashion and be confident of the expected outcome, he must be empowered to understand with whom he is dealing with, and so on. In the section "The Dance of Identity—Implemented by WS-*," we described in detail how the two most common scenarios in the Identity Metasystem are implemented via web services. In those sequences, we have seen how all the negotiations and low-level protocol interactions were performed by an agent. The subject examined the data summarized by the agent and directed its behavior for executing the subject's behavior (for example, disclosing a certain claim to a specific RP). The agent decoupled the subject from the complexities of the underlying system, leveraging all the good properties of the protocols for acquiring as much data as possible and presenting information to the subject in the best way for enabling truly informed decisions.

Windows CardSpace enables Windows users to participate in the Identity Metasystem

Windows CardSpace is the implementation of that agent on the Windows platform. It enables Windows users to participate in the Identity Metasystem, taking care of the nitty-gritty details of RP and IP communications while presenting to the user an intuitive façade. Some form of user agent is a necessity, imposed by the human-integration requirements of the Identity Metasystem. As long as the traffic generated abides to the open protocols we have seen so far and the UI provides an experience compatible with the identity laws, every platform has the freedom to come up with its own (or even more than one) agent. Apple Macintosh and Novell Linux are examples of non-Windows platform for which a user agent is already available. Although there are no guarantees that the experience will be replicated verbatim on every selector and on every platform, thus far the card metaphor is being consistently used across the various projects.

CardSpace presents the user with an easy metaphor, but under the hood it uses the full power of WS- and the Identity Metasystem*

CardSpace is what allows Windows users to experience situations like the wine seller example in an extremely natural fashion, where having your age verified is as simple as clicking a picture of your driving license on the screen. Where the user experience is just natural gestures and the control that derives from it, the system supplies all the intelligence necessary for probing services for policies or calculating what it takes for obtaining a token from a certain STS. Windows CardSpace is intimately tied to its platform. Whereas the traffic it generates is entirely based on the WS-* open standards we mentioned, and hence virtually indistinguishable from the output of user agents on other platforms, the user experience and operating system integration take full advantage of Windows peculiarities and security features. The experience has been designed for the ground up for abiding by the identity laws.

The remainder of the book is dedicated to exploring how CardSpace puts the user in control of his own destiny, by fully leveraging the possibilities offered by the Identity Metasystem.

Summary

Where Chapter 1 described problems and shortcomings, this chapter gave hope about the existence of a sustainable solution.

We started by stressing the need for reaching a solution that would satisfy all online players. We went on exploring the current thinking about identity systems, showing how past errors and success stories were distilled through an industry-wide dialog on the seven laws of identity. We introduced the Identity Metasystem, an abstract model that addresses the common issues of identity management in full respect of the identity laws. We have seen how the Identity Metasystem is not an alternative proposition to today's technology, but rather a further level of abstraction that relies on current systems and facilitates interoperability. Such a design choice guarantees investment protection and makes the solution future-proof, gracefully accommodating yet-to-be invented protocols.

We spent a fair amount of time on the WS-* specifications, understanding their role in the industry and digging into the details of the standards that are more relevant to the identity space. Once we gained more practical knowledge of web services, we were finally able to put all the pieces together and define a solid architecture for the Identity Metasystem model.

After the protocol aspects were all addressed, we defined the role of CardSpace as the user experience designed for empowering Windows users to be first-class citizens of the Identity Metasystem.

This chapter concludes Part I of the book, devoted to understanding the problem we are trying to solve, the solution in its entirety, and the intended role of CardSpace in the grand scheme of things. The remainder of the book focuses exclusively on CardSpace. What it is, how to use it, and how to design systems that take full advantage of it. Part II introduces the technology and the basic use cases from the user and developer viewpoints. Part III then goes into more depth about what it means to be an RP or an IP.