Zero Footprint Secure Internet Authentication using Network Smart Card

Asad M. Ali

Smart Card Research, Axalto.
8311 North FM 620 Road, Austin, TX 78726, USA
amali@axalto.com

Abstract. This paper describes the motivation and technological innovation of Network Smart Card, a next generation smart card architecture that supports standard Internet communication and security protocols. It outlines the role of these next generation smart cards in addressing some of the weaknesses inherent in current Internet authentication frameworks. The paper evaluates several common methods of authenticating users as well as servers during online transactions and shows how they can be improved by the use of Network Smart Card. Traditional two-factor authentication techniques require modifications to client machine, remote server, or both. This paper describes a method of achieving the same two-factor authentication for secure Internet access without requiring any modification to host device or remote servers. Finally, the advantages of Network Smart Card are evaluated against other forms of authentication, such as conventional smart cards and OTP tokens.

1. Introduction

A fundamental assumption behind online transactions over the Internet is an implicit trust in the identity of the other party, and the confidentiality of the data being transferred. This trust is not gained through personal interactions, but is instead based on a combination of underlying technologies that provide authentication and data encryption. In today's world of ubiquitous network access, where ability to conduct secure online transactions is expected of most, if not all, merchants, there is an ever-increasing need to strengthen these trust enabling authentication technologies without degrading the user's online experience. Since attackers continuously exploit weaknesses in existing modes of authentication, frameworks such as simple password based login systems, which were once considered secure, are no longer adequate [1]. In this perpetually changing landscape of online security threats, next generation smart cards could be extremely useful tools. Conventional smart cards [2] have long been used as secure identity tokens for gaining access to local resources. By supporting standard Internet protocols for communication (TCP/IP) and security (SSL/TLS), the next generation smart cards [3] can bring the same level of security to online transactions. The challenge is to transfer these login credentials from the token to a remote online server without requiring a host application, or modifying the remote server. This paper describes one approach to meet this challenge.

2. Network Smart Card

To understand the Network Smart Card architecture we first need to look at the current usage of smart cards. Smart card technology has been in use for more than two decades. However, because of the ISO 7816 based standards [4] embraced by the smart card industry, smart cards have evolved in their own niche markets using protocols that are alien to the mainstream computing world. Although there have been key pioneer attempts [5,6,7,8] to break away from smart card specific standards, the use of smart cards as Internet access tokens is still on the fringes of mainstream computing. Smart card advantages such as security, portability, wallet compatible form-factor, and tamper resistance make them increasingly useful in a wide variety of environments such as GSM. However, in other environments such as desktop computing and online access, wide scale adoption of smart cards is hindered by the mismatch between smart card communication standards and the standards of the mainstream computing and networking. When smart cards are connected to host computers, applications cannot communicate with them using standard mainstream network interfaces. Instead, smart card specific hardware and software in the form of reader device drivers and middleware are needed to access smart card services.

2.1 Motivation

Applications for conventional smart cards, using ISO 7816 based communication standards, are difficult to develop, and even more cumbersome to deploy and maintain. These deployment hurdles have hindered the acceptance of smart cards as secure devices for Internet commerce more than any other single factor [9].

Smart Cards are extremely useful hardware tokens that provide a secure repository of confidential data. However, a conventional smart card cannot guarantee data security beyond its physical boundary without trusting the host device to which it is connected, or sharing a smart card specific encryption technique with the remote application. These restrictions have limited the appeal of smart cards as secure Internet access devices. Data could be manipulated on the host computer before being forwarded to a remote trusted merchant. While merchants may have very high confidence in data retrieved directly from a smart card, they cannot put the same level of trust in the data forwarded by a host computer. Storing data securely is one thing; using it safely is another. While conventional smart cards provide very strong secure storage, safe use is limited to trusted terminals; there is no mechanism to pass data securely to a standard remote server through an untrusted host or terminal using mainstream communication protocols. The Network Smart Card has been designed to overcome this limitation, so that a remote server can trust data from the smart card.

2.2 Architecture

The Network Smart Card solves the communication mismatch by implementing standard Internet protocols on the card. What distinguishes the Network Smart Card technology from some earlier attempts [6,8] at making the smart card network aware, is an architecture based on certain key design choices: implement the TCP/IP network stack and SSL/TLS security layer inside the card; use standard interfaces and drivers that are built into most operating systems, so that no additional middleware deployment is required [3]. This provides seamless connection to host computers and an end-to-end secure data connection with other remote standard Internet nodes.

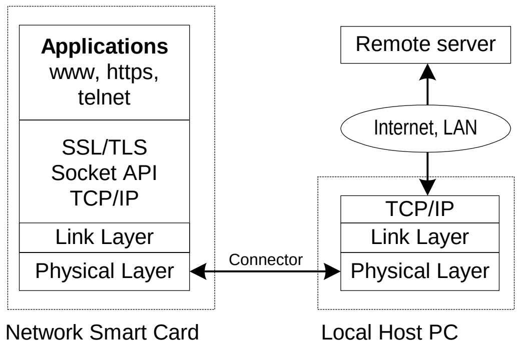

Figure 1: Protocol stack on Network Smart Card

Figure 1 shows an overview of the Network Smart Card architecture. It contains a USB or ISO 7816 physical layer, a complete network stack consisting of a data link layer, TCP/IP, and SSL/TLS, and various network applications. The data link layer can be either Ethernet/EEM or PPP [10]. If a USB interface is used, then a USB connector is used to connect to the host computer. If an ISO 7816 interface is used, a specialized smart card reader is used to convert this to full duplex serial or USB, and is connected to a serial or USB interface on the host computer. The host computer can be any platform that is configured to permit network access from a serial or USB port. This includes most workstation, desktop, and laptop platforms including Windows, MacOS X, Linux and Unix platforms, as well as some mobile palmtop and handset devices. The host is unaware that the computer being connected is a smart card; it treats the smart card as any other computer requesting a direct connection. No middleware or other smart card specific software is required for any platform. This is an enormous leap in the evolution of smart card design and deployment.

The host computer functions simply as a router to connect the smart card to the network. This enables the smart card to access network resources and provide its services through the network without requiring any middleware on the host computer. The remote computers that the smart card communicates with are also unmodified, with no middleware or smart card specific software. As far as the remote computer can tell, the smart card is just another standard computer on the Internet. The ability to establish an end-to-end secure connection with a remote merchant server from any PC can turn the smart card into a portable and secure token for Internet authentication.

3. Current Authentication Methods

The HTTP protocol [11] used on the Internet is a stateless, unencrypted and unauthenticated protocol. What makes it secure for online commerce is the SSL [12, 13], or its IETF flavor TLS [14], protocols. These protocols use public-key cryptography to exchange a secret key and then symmetric cryptography to encrypt the actual data exchanged between a web browser and a web server. However, even with the use of TLS, the security of authenticating users is far from ideal [15,16]. The problem is not with the protocol itself, but with the infrastructure in which it is used.

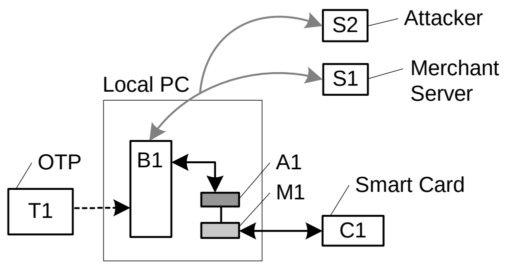

Figure 2: Common Internet authentication frameworks

Figure 2 illustrates various methods of authenticating the identity of users during an online transaction. B1 is a web browser through which a user connects to a merchant server S1. These methods are discussed in the following sections.

3.1 Password

A username/password pair is perhaps the oldest method of identifying a user at a remote server. This single-factor authentication is now universally considered weak for transactions of high value. Although still widely used by most merchants and financial institutions due to lack of a low cost alternative, authentications based solely on passwords have some inherent shortcomings. Good passwords are difficult to remember, while poor ones are easily compromised using various forms of software attacks. Regardless of the password strength, a password typed manually into a browser is vulnerable to a keystroke logger attack before it is encrypted using TLS. In addition, phishing and spoofing attacks may trick the user into providing the password directly to the attacker.

3.2 Automated Password

Automated password entry is achieved using various form-fill applications, A1 (figure 2), that automatically enter the required fields in a web form. This allows use of much stronger passwords that do not have to be remembered by the user. While this approach prevents the typical keystroke logger attack, the data is still kept on the host computer and can potentially be stolen through browser exploits and Trojan horses. It is also vulnerable to spoofing and phishing attacks [17]. Furthermore, this approach is not portable. It can only be used from computers that have application A1 installed and on which the user has previously saved all password data.

3.3 Conventional Smart Cards

A safer, and more portable option is to store the password data on a conventional smart card, C1. Smart cards are extremely secure hardware tokens and are excellent repositories for highly confidential information. They provide a two-factor authentication that is missing in simple password based options. However, due to the mismatch between ISO 7816-based smart card communication standards [4] and the communication protocols [10, 18, 19] used by mainstream PC applications, smart cards require special reader drivers as well as middleware application, M1. This overhead is a major impediment to the widespread use of smart card based authentication solutions.

3.4 OTP Tokens

OTP tokens are small portable devices that generate a one-time-password code, which can be combined with traditional username/password method to provide a two-factor authentication. This alleviates some of the problems associated with password-only methods. There are two broad categories of OTP generation algorithms: time based algorithms such as the one used with RSA SecureID tokens; and event-based algorithms such as that proposed by the Open Authentication (OATH) consortium. While the latter algorithm is an open standard, the former uses a proprietary technology. Regardless of which option is chosen, the token and the authentication server have to be synchronized. Unit cost of OTP tokens is generally higher than that of smart cards. In addition the cost of token distribution and integration of OTP algorithms in the authentication servers have to be evaluated by merchants when opting for this technology.

3.5 Server Authentication

Like user authentication, there are issues with the server authentication as well. For example, the TLS protocol relies on digital certificates to ensure the identity of the two parties. While client authentication is optional, TLS requires the server side to be authenticated before TLS handshake can proceed. This design reflects the general principle of business-to-consumer Internet commerce. It is more important for a user (the client in a TLS handshake) to ensure that the server (typically an online merchant) is who he says he is. After all, securing a connection with the wrong, presumably malicious, merchant can be disastrous. We may like to assume that TLS provides a magic bullet to ensure that we communicate with the correct merchant, but it does not. All it ensures is that we are communicating securely with somebody [20]. To verify that that somebody is actually the correct merchant is left at the discretion of the user. Unfortunately most users do not know how to perform this validation, and of those who know, the vast majority does not make the effort.

Web browsers do assist in this validation process. They compare the intended URL with the corresponding value in the digital certificate received from the merchant. This value is usually the Common Name part of the Subject field in the x.509 certificate that complies with the ITU-T X509 international standard [21]. In case of a mismatch, the browser displays a warning dialog box. The dialog box is also displayed if the certificate has expired, or if the browser does not recognize the Certificate Authority that issued the certificate. These last two warnings are rather rare. Usually, it is the first warning about mismatch of intended URL and actual certificate holder that users overlook. As illustrated in figure 2, a user may think that he is communicating securely with a genuine merchant S1, but in reality could have a secure TLS connection with attacker S2, due to a redirection attack, or simply by mistyping the URL, thereby connecting to a spoofed server.

4. Cardholder Verification

Before the Network Smart Card can transfer the user login credentials to remote servers, the user must first be authenticated to the smart card. To achieve this authentication, the user opens a web browser and connects to the web server running on the Network Smart Card. The web server sends a login page into which the user types his PIN. However, a conventional PIN based mechanism where the PIN is typed into a text box opens the possibility of the PIN being compromised. This is particularly true when using public computers that may have malware and keystroke loggers installed on them. The challenge is to enter the PIN on such systems without compromising the PIN.

We use a methodology whereby PIN or password based user authentication can be achieved without compromising the PIN or password as they are entered by the user. There are two mechanisms used, depending upon the balance of ease of use verses security.

The simplest mechanism uses digitally scrambled numeric images at random locations, which are then clicked by the user to show knowledge of the PIN. When more security is required, a second mechanism is added, through which the user applies a mathematical transformation to the PIN, P, using a transformation PIN, T. This transformation is keyed from a random number, R, which is displayed on the login page. The result of this transformation is a virtual PIN, V, which is a one-time password. P and T are secret numeric values known to the user. R is generated by the smart card and is different for each login attempt. The combination of these three values and the transformation logic can produce a virtual PIN, V that is different each time the user logs in.

The transformation logic can vary in complexity depending upon the security requirements or the comfort level of the user. The logic can also be designed in such a way that the selection of a particular transformation PIN, T, can nullify the transformation effect. In this case the virtual PIN, V, is the same as the actual PIN, P.

5. Zero Footprint Authentication

Once the user has been authenticated to Network Smart Card, the smart card can act on user's behalf and securely send his login credentials to remote online servers. The technique described in this paper is a zero-footprint technique; requiring no change to either the host computer or the remote servers. It uses JavaScript and a standard web browser on the host computer to transfer login information from Network Smart Card to a remote unmodified server. No additional application software is required on the host. Furthermore, since the Network Smart Card does not require any smart card specific middleware or reader drivers, this approach provides an extremely portable way of carrying the login credentials of multiple existing commercial servers on the smart card and then logging into these servers.

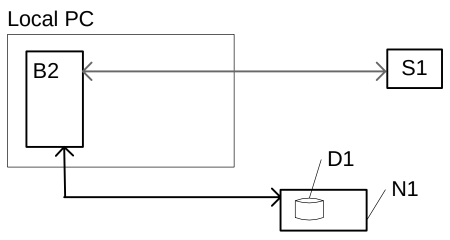

Figure 3: Auto login from smart card without A1 or M1

Figure 3 shows a high-level overview of this approach. It is a simple design where a browser B2 connects to the login page of the remote server S1. The user login data is passed from the smart card N1 to the remote server S1 via browser B2. There is no need to have the application A1 or the middleware M1 installed on the host computer.

Figure 4 provides a more detailed view of how the Network Smart Card can be used to achieve this. All arrows indicate data flow over a secure HTTPS connection in response to user clicks or automated script processing.

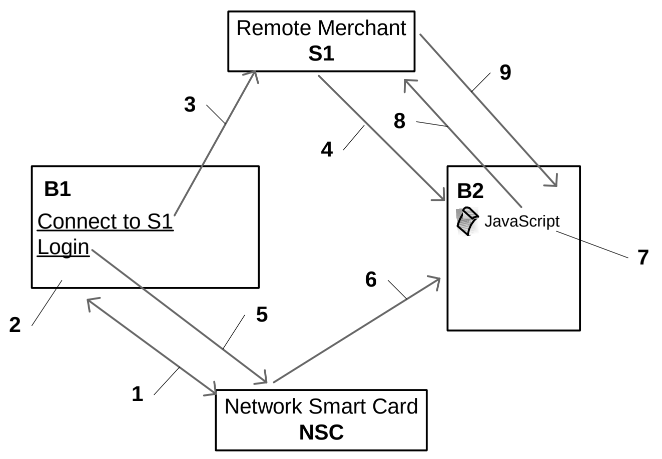

Figure 4: Steps for passing login data from smart card to remote server

The description of each step in Figure 4 is as follows:

1. The user opens a browser B1 and connects to the web server running on the NSC [Network Smart Card]. One way to do this is to type the IP address of the card in the URL bar of the browser. After entering the PIN the user is authenticated to the card.
2. The NSC sends a page to browser B1. This page has the list of remote servers for which the card has login credentials. Each server is represented by a pair of HTML links; one connecting to the login URL of the remote server, and the other back to the NSC.
3. The user clicks on one such server link, e.g. server S1. A connection is made to the login URL of the server.
4. The login page is downloaded from server S1 into a new browser window B2. This allows the server to write any cookies that are necessary during the login process. The cookies are written to the host PC. In case these cookies are session specific they are associated with the browser instance B2.
5. Instead of filling the login information in B2, the user now clicks on the corresponding login link in B1. This link is the second half of the pair of server links picked in step 3. The click sends a request back to the web server on the NSC.
6. The response from the NSC consists of an HTML form template that matches the form used for login at server S1. Along with this form, a small JavaScript code is sent. The form data as well as the JavaScript code are loaded in browser instance B2. This is the same browser that previously contained the login page from server S1. All the form elements are marked as “hidden” so they do not show up in the browser window. Instead a message is displayed indicating that user login information is being sent to server S1.

7. The JavaScript code is automatically launched. It uses data from the NSC to fill the login form, and then calls the submit( ) action on the form.
8. The form containing user login information is sent to the URL on server S1 that authenticates login requests.
9. Once server S1 validates the login credentials sent in step 8, the user is granted access.

This allows the NSC to connect seamlessly to an unmodified remote server. The two-step login procedure that requires the user to click twice (step 3 and 5), though somewhat cumbersome, is designed to work across most common web browser and operating system combinations. It is possible to achieve the same zero-footprint login functionality with a single click if we restrict the browser and OS platform options.

6. Authentication Details

The procedure to authenticate a user to remote unmodified web servers uses a combination of HTML form data and JavaScript code to transfer user login credentials from a Network Smart Card to the target server.

6.1 Simple Ideal Case

Theoretically, a user login name and password can be sent to a web server using a simple HTML form template. This form template has three elements that are of interest:

- The target URL at the authentication server to which the form data has to be posted, e.g. *https://www.serverS1.com/doLogin*
- The name of the input tag corresponding to the username, e.g. *userID*
- The name of the input tag corresponding to the password, e.g. *userPassword*

Once these three elements are known, a form can be constructed as follows:

```
<form action="https://www.serverS1.com/doLogin">
     <input name="userID" value="myUserID">
     <input name="userPassword" value="myUserPassword">
</form>
```

Figure 5. HTML Form template for sending user login data

The actual values for the *userID* and *userPassword* fields can be stored on a secure hardware token like a Network Smart Card. These values are then read from the smart card and placed in this form template. The form is then submitted to the URL indicated in the action element. In theory this is all that is needed to login to the remote server. In practice this approach seldom works.

6.2 The Real World

“In theory, there is no difference between theory and practice. But, in practice, there is” [22]. The reason the simple theoretical case described in section 6.1 does not work in the real world is the use of session cookies by merchant web servers. Servers store cookies on the local client machines for two reasons. The first is to identify users and keep track of their browser session at the server. The second reason is to prevent repeated automated login requests. Servers regard such requests as attempts by a potentially malicious user to break into existing accounts on the server. Therefore, servers reject login requests from browsers that fail to present an adequate set of cookies.

These cookies are written to user’s machine when the browser connects to the login page of the server. The server can choose to put one or more cookies for each login session. The cookies can also be time stamped so they cannot be used after their predefined validity period has expired. All this is done to make sure that it is an actual user who is trying to login, and not an automated script with malicious intentions. If the login form is sent directly from the Network Smart Card the corresponding session cookies will be stored on the card itself. The remote server will not recognize the session with a web browser running on the host computer since this browser will not have access to these cookies.

The challenge then is to make the simple ideal case scenario work in the real world environment for transferring data from a smart card to the remote server.

6.3 The Solution

One solution is to perform the login process in two steps. This solves the disparity between the simple form submission scenario and the real world authentication environment where commercial web servers use an extensive set of session cookie logic. In the first step the login page from the remote server is downloaded in a browser B2. The remote server is thus given the opportunity to write all authentication related cookies to the local host machine. Once the cookies are written, the same browser instance can be used to load and send the HTML form template of figure 5 to the server. Now the remotes server accepts the login credentials passed by the form and the user is granted access.

JavaScript can be used to automate part of this two-step login process so that the end user can login to unmodified servers with just two clicks; one click to connect to the login page of the server, and second click to pass the user data to the server

6.4 A Complete Example

This section explains the details of HTML and JavaScript code that can be used to pass login data from a smart card to a remote merchant server. Figure 4 is described again, but with actual code examples.

Step 1: The user opens a browser B1 and connects to the IP address of the NSC [Network Smart Card]. After entering the PIN the user is authenticated to the card.

Step 2: The NSC sends a services page to browser B1. This page has the list of remote servers for which the card has login credentials. Each remote merchant server is represented by a pair of links. The HTML code for one such pair of links is shown in Listing 1.

```
1   <HTML><BODY>
2   <A href="https://www.serverS1.com/login" TARGET=B2> Connect to serverS1 </A>
3   &nbsp;
4   <A href="https://myNetworkSmartCard/serverS1.html" TARGET=B2> Login </A>
5   </BODY></HTML>
```

Listing 1: Pair of links to login to a merchant server S1

The key aspects of this code are:

- The link on line 2 connects to the login page of the remote merchant *server S1*.
- The link on line 4 connects to the corresponding page on the smart card that has login data for the merchant server S1. Both the links have browser B2 as their target.

Step 3: The user clicks on one such server link, e.g. *server S1*. A connection is made to the login URL of the server, https://www.serverS1.com/login.

Step 4: The login page is downloaded from *server S1* into a new browser window B2. This allows the server to write any cookies that are necessary during the login process. The cookies are written to the host PC. In case these cookies are session specific cookies they are associated with the browser instance B2.

Step 5: Instead of filling the login information in B2, the user clicks on the login link (line 4, Listing 1) in B1. This is the link corresponding to the server link picked in step 3. This click sends a request (https://myNetworkSmartCard/serverS1.html) back to the NSC web server.

```
1   <HTML><BODY>
2   Secured by Axalto Network Smart Card :<br>
3   Your login credentials are being passed to server S1, please wait ...
4   <FORM method="post" name="loginForm" action="https://www.serverS1.com/doLogin">
5   <INPUT type="hidden" name="userID" value="">
6   <INPUT type="hidden" name="userPassword" value="">
7   </FORM>
8   <IFRAME SRC=serverS1Login.html name="autoLogin" ALIGN=bottom FRAMEBORDER=0>
9   </IFRAME>
10  </BODY></HTML>
```

Listing 2: serverS1.html file on Smart Card.

Step 6: The response from the NSC is displayed in browser B2, over-writing the login page from server *S1*. This response data consists of an HTML form template that matches the form used for login at *server S1*. The HTML code is shown in Listing 2.

The key aspects of code in Listing 2 are:

- Line 2 and 3 show a message that is displayed to the user while login data is being sent to *server S1*. This is the only text visible to the user. All other data is hidden.
- Line 4 is the start of a hidden form. The action element of the form is set to the URL at server S1 that processes login requests.
- Line 5 is the input element for userID at server S1. It is hidden.
- Line 6 is the input element for user's password at server S1. It too is hidden.
- Line 8 creates an inline frame on the same page. The source of this page is another HTML file, serverS1Login.html on the smart card. The code for this file is shown in Listing 3. It contains the JavaScript code to fill the username and password data and to automatically submit the form to server S1.

```
1   <SCRIPT LANGUAGE="JavaScript">
2   function setValue() {
3        parent.document.loginForm.userID.value = "myUserID";
4        parent.document.loginForm.userPassword.value = "myUserPassword";
5        parent.document.loginForm.submit();
6   }
7   </SCRIPT>
8   <BODY OnLoad="setValue()">
9   </BODY>
```

Listing 3: serverS1Login.html file on Smart Card.

The key aspects of code in Listing 3 are:

- Line 3 and 4 set the values of username and password.
- Line 5 submits the form to server S1.
- Line 8 indicates that the function setValue() should be called as soon as this frame is loaded. This allows the login data to be submitted automatically.

Step 7: The JavaScript code from serverS1Login.html is automatically launched (see line 8 Listing 3). The confidential user data is kept on the smart card and placed in serverS1Login.html file before it is sent to the browser B2.

Step 8: Browser B2 sends the form data containing user login information to the URL (specified in line 4 of Listing 2) at server S1 that processes login requests.

Step 9: Once *server S1* authenticates the user, user is granted access.

This example described the use of HTML and JavaScript code to pass user login data to the merchant server. The key is to reuse the same browser instance B2 through which session cookies were initially obtained. This browser reuse is possible through the TARGET element of HREF tag. Another option is to use browser window handle. The first link (that goes to the login page of server S1) creates a window. The handle of this window is saved inside the JavaScript code. The second link (that goes to serverS1.html on smart card) reuses this window handle.

7. Comparison

Table 1 compares the security, portability, and deployment cost of various authentication methods. The deployment cost is measured in terms of required changes on both the client machine from which the user logs in, and the remote servers through which the user is authenticated. As shown, the conventional smart cards require no change at the server, but need installation of middleware application and smart card reader device drives on the client machine. Conversely, the OTP based solutions require no change to the client machine, but the OTP algorithm needs to be synchronized with authentication servers. In contrast, the Network Smart Card based solution proposed in this paper requires no change at either the client or the server. It provides a two-factor authentication for passing user identity to existing commercial servers. These servers are not necessarily aware that the identity credentials are coming from a smart card. The only assumption implicit in this approach is that the authentication server does not change its login URL and form template tags. If any one of these things is modified, a corresponding change is required to the login scripts stored on the Network Smart Card. Since the cost of server modification is often an impediment to achieving a stronger authentication framework, this seamless integration with existing servers can be an attractive alternative to conventional smart card or OTP token options.

Table 1. Comparison of various authentication methods

Authentication Methods	**Security**	**Portability**	**Requires Change at:**	
			Client	**Server**
Manual Password	Low	High	No	No
Auto Form-fill	Medium	Low	Yes	No
OTP Token	Medium	High	No	Yes
Conventional Smart Card	High	Low	Yes	No
Network Smart Card	High	High	No	No

8. Progress

This technology was first prototyped and demonstrated as a server at Cartes 2003. Client technology was added in 2004. A second-generation prototype that allows smart card assisted login to unmodified remote online servers, was completed in 2005, and demonstrated at Cartes 2005.

9. Conclusion

This paper presented the Network Smart Card architecture and outlined its role in breaking the restrictive mould of conventional smart cards. This new technology recasts smart cards as full-fledged network aware devices that can enhance the trust and security of online authentication frameworks. Network Smart Card can be used for passing user login credentials to remote web servers to gain access to existing site. The user data is kept on the smart card and is sent to the remote server through a secure SSL/TLS connection. This technique has advantages over other existing form-fill software approaches. It provides a more portable way of passing login data from any machine. The user is not restricted to the machine on which form-fill software is installed. The technique is also superior to general OTP-token based methods. It does not require any change on the server side and can, therefore, be used seamlessly with existing commercial servers. In addition, since URLs of legitimate merchant web sites are also stored on the smart card, the user can be protected from potential phishing, spoofing, and DNS poisoning attacks. Since Network Smart Card is a secure computing device supporting standard mainstream communication and security protocol stacks, it is in a much better position to prevent such attacks than conventional smart cards or OTP tokens.

References

[1] B. Schneier, “Secrets and Lies: Digital Security in a Networked World”, pp. 17-39, ISBN 0-471-25311-1, Wiley Computer Publishing, 2000.

[2] Jurgensen, T.M. and Guthery, S.B., “Smart Cards”, Pearson Education, Inc., 2002.

[3] Montgomery, M., Ali, A., and Lu, K. "Implementation of a Standard Network Stack in a Smart Card", CARDIS 2004, Toulouse, France, August 2004.

[4] ISO/IEC 7816-3:1997 “Information technology – Identification cards – Integrated circuit(s) cards with contacts – Part 3: Electronic signals and transmission protocols”. Available from International Organization for Standards; http://www.iso.org.

[5] Rees, J., and Honeyman, P. "Webcard: a Java Card web server," Proc. IFIP CARDIS 2000, Bristol, UK, September 2000.

[6] Urien, P. “Internet Card, a smart card as a true Internet node,” Computer Communication, volume 23, issue 17, October 2000.

[7] Guthery, S., Kehr, R., and Posegga, J. “How to turn a GSM SIM into a web server,” Proc. IFIP CARDIS 2000, Bristol, UK, September 2000.

[8] Muller, C. and Deschamps, E. “Smart cards as first-class network citizens,” 4th Gemplus Developer Conference, Singapore, November 2002.

[9] J. Vijayan, “Low Draw for Smart Cards: Cost and interoperability problems are slowing companies' adoption of smart card technology”. ComputerWorld, February 2004. www.computerworld.com/printthis/2004/0,4814,89924,00.html

[10] Simpson, W. “The Point-to-Point Protocol (PPP),” RFC 1661, July 1994.

[11] Fielding, R., et al. “Hypertext Transfer Protocol -- HTTP/1.1” Network Working Group, RFC 2616, June 1999. The RFC is available at: http://www.w3.org/Protocols/rfc2616/rfc2616.html

[12] Freier, Alan O., et al. “The SSL Protocol, Version 3.0,” Internet Draft, November 18, 1996. Also see the following Netscape URL: http://wp.netscape.com/eng/ssl3/.

[13] Elgamal, et al. August 12, 1997, “Secure socket layer application program apparatus and method.” United States Patent 5,657,390.

[14] Dierks, T., Allen, C., “The TLS Protocol, Version 1.0,” IETF Network Working Group. RFC 2246. The RFC is available at, http://www.ietf.org/rfc/rfc2246.txt.

[15] Jesdanun, A., “Thief captures every keystroke to access accounts,” Seattle Post, July, 2003, http://seattlepi.nwsource.com/national/131961_snoop23.html.

[16] Poulsen, K., “Guilty Plea in Kinko's Keystroke Caper,” SecurityFocus, July 18, 2003. http://www.securityfocus.com/printable/news/6447.

[17] Poulsen, K. “California reports massive data breach” SecurityFocus, October 19, 2004. http://www.securityfocus.com/news/9758

[18] Postel, J. “Internet Protocol,” RFC 791, September 1981.

[19] Postel, J. “Transmission Control Protocol,” RFC 793, September 1981.

[20] B. Schneier, “Secrets and Lies: Digital Security in a Networked World”, pp. 167-168, ISBN 0-471-25311-1, Wiley Computer Publishing, 2000.

[21] X.509 certificate standard from International Telecommunication Union (ITU-T). See http://www.itu.int/ITU-T/index.html for a copy of the standard.

[22] Jan L. A. van de Snepscheut (1953-1994), Computer scientist and educator.