User Authentication for Online Applications Using a USB-Based Trust Device

Julian Jang, Dongxi Liu, Surya Nepal, and John Zic

CSIRO ICT Centre, PO Box 76, Epping NSW 1710, Australia
{julian.jang,dongxi.liu,surya.nepal,john.zic}@csiro.au

Abstract. We present a system that enables secure user authentication by leveraging a portable USB-based trusted device. The heart of our system runs a protocol which guarantees trusted behavior at multiple layers; from the hardware device itself, to the software executing on the hardware, and finally to the application hosted in the remote server. This combination assures end-to-end trust and makes our system resilient to physical attacks (e.g. to the device and wire tapping) as well as logical attacks (e.g. main-in-the-middle attack). Our system utilizes web-based proxy communication using standard HTML tags and JavaScript to coordinate communication amongst different components. This enables our system not having to install any extra drivers typically required for supporting communication in most existing technologies.

Keywords: user authentication, trusted computing, trust device, web communication.

1 Introduction

There is a growing trend to provide services online. At the core of online applications is the need for user authentication. The enterprises providing online services must determine whether a user is, in fact, who he or she claims to be by verifying the proof of knowledge presented as a form of user credentials (e.g. user secrets) [3]. Cyber criminals have targeted the attacks on these user credentials to gain economic benefits.

Predominantly, the attacks on stealing user credentials happen at two different levels [9]. One level of attack happens at the user's platform. The criminals attack directly to the user's platform by exploiting the inherent problems with insecure device. This has led to the development of separate hardware tokens using a micro-processor platform, such as smart cards or Trusted Platform Module (TPM) [22], to use them as secure devices that store the user credentials in a tamper-resistant way.

Another level of attack occurs at the data transfer channel when the attackers directly intercept user credentials exchanged between the client platform and the server machine. Authentication based on public key infrastructure (PKI) has known to rectify the problem. The user authentication based on the combination of hardware token and the PKI has been explored. We present existing state-of-art of these two combination and their limitations.

The most successful and widely adopted user authentication mechanism based on the combination of hardware token and PKI is from the smart card space. The smart card technology implements a variety of hardware and software countermeasures that can thwart attacks against the card itself (e.g. the card acting as a secure device) and the software infrastructure (e.g. authentication of card user via a variation of using PKI). However, it is reported [9] that the attackers could intercept a private identification number (PIN), which is used to unlock the smart card before private-key related functionality takes place, entered to the smart card via the PC. Another drawback of using smart card technology is that it requires a specially designed device to read the smart card such as a keyboard with a slot to slide the card in.

Cell-phone has gained its popularity as hardware tokens as it allows users to utilize their mobile gadgets (i.e. mobile phone and PDAs) as a part of authentication tool. Phoolproof by Parno et al [18] and MP-Auth by Mannan et al [14] are two such examples. Both approaches use a cell-phone as their hardware platform that stores user's secret then use variation of PKI based authentication to authorize the mobile user. Though, the trustworthy of cell-phone remains stronger than usual PCs, more and more malware are targeting cell-phones. Worms such as Cabir[6] are designed to spread in smart phone by exploiting vulnerabilities in embedded operating systems used on smart phones. Regular cell-phones with J2ME MIDlets have also been targeted by RedBrowser Trojan [7].

USB-based trusted devices are gaining the acceptance as the next generation of hardware token due to the cheaper cost and ubiquitous USB ports in most PC and laptops. Many state-of-art USB-based trusted devices today [1,4,8,10,13] contains a customized OS, a high performance processor, and integrated flash memory card. These allow the token to accommodate many desired software (both API and runtime executables) and to have storage capacity. As the capacity of the token become more powerful, the requirement for the token to communicate across the Internet, possibly via the host computer, has been raised. And many existing solutions today ask the users to install extra software to enable such communication. However installing extra software has been seen as burden rather than helping to many end users.

This paper presents a system that provides a degree of a trust by securely authenticating the user to the server by using hardware-based PKI mechanism and driver-less installation of USB-based trusted device. Our system leverages the capabilities of a USB-based trust device Trusted Extension Device (TED) [17] to provide temper-resistant protection to the device itself and to assist in secure operations. Our system runs a PKI-based remote attestation protocol which guarantees trusted behavior at the hardware device itself, to the software executing on the hardware, and to the application hosted in the remote server. This combination assures end-to-end trust resilient to physical attacks as well as logical attacks. Our system utilizes web-based proxy communication using standard HTML tags and JavaScript to coordinate communication amongst different components. This enables our system not having to install any extra drivers typically required for supporting communication. We have implemented our protocol along with the overall system to demonstrate the viability of our solution.

The paper is organized as following. In Section 2, we discuss our system design principles. In Section 3, we describe our system architecture; this is followed by our protocol and its analysis against threat models in Section 4. In Section 5, we describe our prototype system and its performance. We present concluding remarks and the future works in Section 6.

2 System Design

2.1 Technological Foundation: Trusted Computing Technologies

There are two important aspects of trusted computing technology [21] that we utilize in our system and protocol: Trusted Platform Module (TPM) [22] and Remote Attestation. The TPM is a microcontroller system specified by the Trusted Computing Group (TCG). At the time of manufacture, a cryptographic key pair, known as the Endorsement Key (EK), is generated and stored inside the TPM chip. The private part of the EK is held securely by the chip, and is never exposed. Remote Attestation is a method to prove to a remote computer that messages are from TPM-enabled trustworthy computing platform. Somewhat tied to a TPM's EK, is an Attestation Identify Key (AIK) which is created during attestation for use by a particular application. The public part of the AIK is certified by an appropriate trusted third party call "privacy CA" as being the key of a particular TPM. Using the private part of the AIK, which resides only inside TPM and is never transmitted to any external components, the TPM measures its configuration, known as the Platform Configuration Register (PCR) value, and reports this to the remote computer. The remote computer can verify that those assertions, namely the AIK certificate and PCR value, have come from a genuine TPM. We have a previous work on improving Remote Attestation to suit better in the Internet environment [12].

More recently, we proposed a portable Trust Extension Device (TED) [17]. Rather than embedded into the motherboard of a PC, our TED implements a TPM chip inside a USB device to enhance mobility and portability. Our TED v.1 described in [17] is using QEMU and software simulated TPM emulator. However, since then we have progressed with our effort producing TED v.2 which uses a real TPM chip inside the USB along with a Linux OS as a high performance processor. The paper on the TED v.2 is forthcoming and the simplified architecture is shown in Figure 1. With our TED, users can carry it around, plug-in it on any host machine using a ubiquitous USB connection, creating its own secure environment. TPM chip inside our TED generates a set of private keys which participates in running cryptographic functionalities and secure communication in conjunction with the preloaded Linux Operating System.

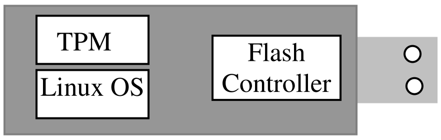

Fig. 1. A high level component overview of TED

2.2 Security Goals

We aim to satisfy following security requirements in our system.

- **Trusted user authentication:** In many security systems including SSL/TSL, authentication mechanisms are provided by a server [15]. Typically server authentication is performed before performing any online operation to ensure the user connects to legitimate server. However, in most cases, the user authentication is naively ignored. The attackers exploit the lack of user authentication mechanism to steal user credential and connect to the server with the victim's credential. In our system, a digital certificate for a user is created as a part of remote attestation protocol. The user digital certificate is then used to authenticate the user before running any transaction with the remote server.

- **User privacy:** It's not uncommon that underlying PC records transaction details and generate customized user profiles. One notable example is cookies. Many times this type of unauthorized PC records is used by cyber criminals to invade user's privacy such as sending unsolicited spam mails. Our system uses a digital certificate that can only be activated by the owner of the TPM but by no one else. Even if somehow the PC manages to steal the TED digital certificate it would not be possible to read the details of the digital certificate to find any details of the user.
- **Trusted path from the user to applications:** Spoofing and "man-in-the-middle" attack both eavesdrops and modifies the data in transfer. These types of attacks have been amongst the most insidious attacks. The trusted path supported by our protocol enables guaranteed input and transfer all the way through the hardware level to the application layer.

2.3 System Design Principles

Following design principles have been considered:

- **Support for heterogeneous platform environments:** Many security solutions fail when they are applied to different platforms. Different solutions are devised for different platforms including device drivers. In our approach, we only need a single version of TED that works with heterogeneous platforms.
- **No extra software installation:** Mobile devices generally require installation of drivers to communicate to the server. This provides a security loophole for cyber criminal in one hand and puts a burden of installation to the users on the other hand. One of upmost design goals in our system is to make the driverless device. We achieve this by using the standard browser technologies such as HTML and JavaScript.
- **No requirement for specially designed device:** Some hardware tokens such as smart cards [20] require a specially designed card readers to unlock the smart card and to access the private data stored in the server. One of our design goals is to free host from having extra infrastructure.
- **Zero footprints:** Cyber criminals always target the vulnerabilities in host machine's hardware and software platform. In our system, we avoid using resources from the underlying platform. Cryptographic keys are stored securely inside TPM chip inside TED and communication is handled strictly following our security protocol leaving no trails of transaction to the underlying platform.

3 System Architecture

Figure 2 shows the system architecture with three components in our system: a client machine and TED, the application server, and a Privacy CA. The components are connected over the Internet and use HTTP to communicate.

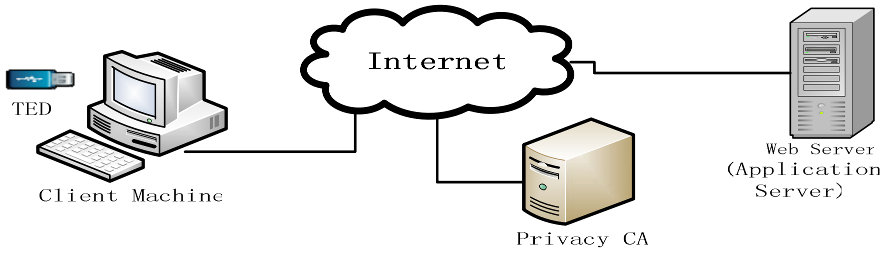

Fig. 2. A system architecture for TED development application and use

3.1 Client Machine and TED

When a TED is plugged into the client machine, it gets its power from the USB port and builds a local IP connection with the client machine after the booting is finished. By default, TED has a single fixed local IP address assigned to it while the client has an IP address allocated dynamically by the DHCP server to connect TED. Using these two end points, TED transmits data to the client machine using HTTP protocol. TED provides the following three operations.

- Collate Identity Key request: TED uses this operation to create a pair of RSA keys called Attestation Identity Key (AIK), the public part of AIK is sent the privacy CA to certify.
- Activate Identity Key request. When TED receives a certificate for the AIK from the privacy CA, it requires running this operation to activate the AIK key as an attestation key. Only the owner of the TPM chip can activate the AIK key since owner authorization is required.
- Sends AIK digital certificate to the server: Newly activated AIK certificate is sent to the application server and is used as a part authentication along with userID and password.

3.2 Privacy CA

Each TED is issued with the credentials including an RSA key pair called the Endorsement Key (EK). The Privacy CA is a trusted third party authentication entity that is assumed to know the credential details along with the public parts of the Endorsement Keys of all TEDs. Whenever a TED needs to communicate with the enterprise, such as the server hosting web applications, it generates a second RSA key pair, called an Attestation Identity Key (AIK). The AIK is sent as a part of identity key certification request to the Privacy CA, which contains, (a) an identity public key, (b) a proof of possession of identity for the private key, and (c) the endorsement certificate containing the TED's endorsement public key. The privacy CA checks the validity of the information in the request. If validation succeeds, the privacy CA returns an identity certificate encrypted with the TED's endorsement public key.

3.3 Application Server

The role of our application server is to host Web application and validate user certificate. Our current prototype system hosts an in-house developed banking application. When TED user enters userID and password via login page our banking server sends a challenge to provide a valid certificate. This is to ensure the userID and password is entered from no other than the legitimate TED user. Once the TED user's certificate is authenticated in conjunction with the corresponding userID and the password, the banking application server let's the TED user to do transactions such as viewing account balance or transferring money from one account to another.

4 System Implementation

4.1 Web-Page Components

One notable contribution of our system is that it enables our TED to communicate to the remote server without having to install any extra software. We achieve driver-less communication using web-based proxy. We utilize the capability of standard HTML scripts that allows multiple frame communication.

Figure 3 illustrates the overall web page components. A JavaScript-based proxy is embedded in the main page. The proxy is basically a listener that watches out any incoming/outgoing messages. Two iframe tags are created within the main page. Each iframe uploads HTML files that are located on the TED and the privacy CA respectively when the main page is rendered by the client browser. Once these HTML files are uploaded, now java Scripts locate on both TED and privacy CA are ready to communicate to the application server via the proxy on the main page.

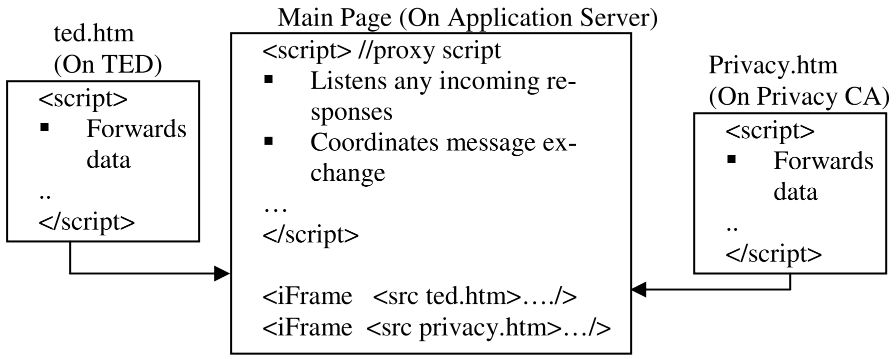
ted.htm
(On TED)
<script>
▪ Forwards data
..
</script>
Main Page (On Application Server)
<script> //proxy script
▪ Listens any incoming responses
▪ Coordinates message exchange
…
</script>
<iFrame <src ted.htm>…./>
<iFrame <src privacy.htm>…/>
Privacy.htm
(On Privacy CA)
<script>
▪ Forwards data
..
</script>

Fig. 3. HTML and JavaScript design for Web-based communication

4.2 The Protocol

Operational assumptions: The goal of the protocol is to protect user's credential from being attacked by malware. The following operational assumptions are made in our protocol. First, we assume that each TED is equipped with an endorsement key which can identify each TED device uniquely. The privacy CA knows the legitimate TED endorsement certificates. Second, we assume that the public part of privacy CA's identity key is publicly available and the TED knows about it. Finally, we assume that the application server maintains a database of legitimate username and password combinations so it can verify upon a login request. Table 1 lists the notations used in our protocol followed by the protocol steps.

Table 1. A list of notation used in our protocol

Notation	Description
T, P, A	TED user, Privacy CA, and Application server respectively
ID, password	TED user's userID and password
AIKpub	Attestation Identity Key (AIK) public key part
PCApub	Privacy CA's public key part
EKpub	Endorsement Key (EK) public key part
S, Q	Secret session keys
N	Nonce is a random number
[]	collection of data
{data}k	Symmetric (secret-key) encryption of data using a secret key k
{|data|}k	Asymmetric (public-key) encryption of data using a public key k

1. The TED is plugged in the host machine, launches a browser on the host machine, then visits the application main page hosted by the remote server machine. The main page contains two iframes;
   ```
   <iframe src=<%=pcaURL%>/privacyCA.htm id="caFrame" />
   <iframe src=<%=tedURL%>/ted.htm id="tedFrame" />
   ```
   When the main page is rendered, two html pages, privacyCA.htm and ted.htm, are uploaded connecting the applications server to the TED and privacy CA. The TED client enters the userID and password and sends them via HTTP message to the application server as a part of a login request. $A \leftarrow T : [ID, password]$
2. Upon receiving a login request, the application server checks whether the userID and password match one of the records in the database of userID and password. Once a match is found, the application server creates a challenge message (i.e. nonce of a random number). The host machine forwards the challenge request to the TED. $T \leftarrow A : [N]$ using the following javaScript-based proxy communication contained in the main page.
   ```
   function sendChallenge(nonce) {
      var win = document.getElementById("tedFrame").
                contentWindow;
      win.postMessage(nonce, "'"+<%=tedURL%>+"'");
   }
   ```

3. The challenge message is received by the eventListner contains in the ted.htm inside TED device.

   ```
   window.addEventListener("message", toTED, false);
   function toTED(e) {
     server = e.origin;
     xmlhttp = new XMLHttpRequest();
     var head = e.data.substring(0, 9);
     var body = e.data.substring(9, url.length);
   }
   ```

   TED creates an AIK key and constructs a special data structure called an IdenityProof. The IdentityProof contains two important data: endorsement key (EK) and identity-binding signature. The EK uniquely verifies a genuine TPM. The identity-binding signature contains a public part of AIK key, user identity, and public part of Privacy CA key which then is signed by the private part of AIK key. The identity-binding signature verifies the newly generated AIK is from a legitimate TPM. TED generates a session key S to encrypt the IdentityProof. The session key S is encrypted by the privacy CA's public key. $P \leftarrow T : [\{Identity \Pr oof\}s, \{| s |\}PCApub]$ This are now sent back to the main page using the script in the main page which is described below.

   ```
   // this script is a part of the function toTED(e)
   xmlhttp.onreadystatechange = fromTED;
   xmlhttp.open("POST",head,true);
   xmlhttp.send(body);

   function fromTED() {
     if (xmlhttp.readyState==4)
       top.postMessage(xmlhttp.responseText, server);
   }
   ```

4. The main page forwards the encrypted IdentityProof to the privacy CA via the javaScript proxy.

   ```
   window.addEventListener("message", function(e){
   ...
         var fr1 = document.getElementById("tedFrame");
         fr1.contentWindow.postMessage(e.data, <%=pcaURL%>);
   ...
   }
   ```

5. The privacy CA receives the encrypted IdentityProof using the javaScript contained in the privacyCA.htm

   ```
   window.addEventListener("message", toPCA, false);

   function toPCA(e) {
     server = e.origin;
     xmlhttp = new XMLHttpRequest();
     var head = e.data.substring(0, 9);
     var body = e.data.substring(9, url.length);
   }
   ```

   The privacy CA uses its private key to decrypt the session key S. Using the session key S, it decrypts the IdentityProof. The EK and identity-binding signature contained in the IdentityProof structure are inspected to verify that the IdentityProof was created by the genuine TPM. After the validation, the privacy CA creates an AIK digital certificate that contains the user identity and signs it to maintain its integrity. The privacy CA creates a session key Q to encrypt the signed AIK digital certificate, denoted as symBlob. To maintain the integrity of the session key Q, the privacy CA creates an asymmetric blob, denoted as asymBlob. The asymBlob contains the session key Q and the hash value of AIK public key. The asymblob is encrypted by the public part of the EK. $T \leftarrow P : [\{symBlob\}q, \{|asymblob|\}EKpub]$ This is sent to the main page using the similar javaScript described in the step 3.

6. The main page forwards the privacy's response to the TED using the similar javaScript used in the step 4.
7. TED uses its private part of the Endorsement Key (EK) to decrypt asymBlob. From the decrypted asymBlob, TED recovers the session key Q and the hash of AIK public key is validated. If AIK public key hash is correct, TED decrypts the symBlob and retrieves the AIK digital certificate using the session key Q. TED sends the AIK digital certificate, denoted as AIKcert, to the application server via the main page using the javaScript descriebd in the step 3. $A \leftarrow T : [AIKcert]$
8. Upon receiving the AIK digital certificate, the application server checks the validity of AIK certificate by verifying it with the public key of Privacy CA. For a valid certificate, the application server checks whether the user identity specified in the AIK certificate matches the userID received earlier in the step 1. If they match, the application server knows that the digital certificate owner is the one who is logged in.

4.2 Threat Models and Security Analysis

We now present a number of threat model scenarios and analyze any security loopholes.

a) *UserID and password hijacking*: It is possible that the TED host machine is compromised by malware. This allows the attacker to be able to intercept UserID and password, perhaps through the hacked browser cookies or monitoring host machine input channels such as keyboards or mouse. Now the attacker tries to access the bank application server using the stolen userID and password. However, the bank application server in our system will request a AIK digital certificate upon log-in request. Since the AIK digital certificate can only be generated by the genuine TED owner using a private part of the AIK key that never leaves the TPM inside the TED. The attacker cannot provide the legitimate AIK certificate and therefore cannot access the private data stored in the application server. However, TED cannot prevent the interception of the screen output such as displayed bank balance to be captured by the attacker.

b) *Malware on TED*: We have made an assumption that our TED. We have made this assumption because the device like TED is most likely manufactured within a tightly controlled environment. For example, for our banking server application example, it will be the bank (via possibly trusted semiconductor company that the bank has a strict contract written on) that actually manufactures and distributes the TED to its customer. When our TED is plugged in, it does not utilize any resources from the TED host machine it is connected. This makes hard, if not impossible, for malware to be installed in the device practically preventing any credential-stealing attacks. Additionally, integrity is measured and stored in the secure storage in the TPM chip inside TED. If this value is altered, AIK cannot be activated.

c) *Theft on TED*: All cryptographic keys in our TED are stored in its temper-resistant secure storage. To unlock any cryptographic key inside the TED and perform any cryptographic functionality, the TED user needs to supply both the owner secret and the password for the storage root key. The attacker must obtain these secrets before the user discovers the theft and revokes the stored keys. It is an unlikely situation that the attackers manage to steal both the device and the secrets.

d) *Man-in-the-middle Attack on the USB channel or the Network*: The attackers launch the channel-breaking attack by eavesdropping the network where the data being transferred. It might be possible for the attackers to intercept the data in transfer. However, the attackers would not be able to read the data. The network between the TED user and the bank application server is protected by SSL. The network between the TED user and the privacy CA is protected by the encryption using a private part of the AIK that never leaves the TPM chip inside TED. Unless the owner of the TED, it is not possible for the attackers to be able to decrypt the data because the attackers will never be able to get the appropriate AIK private key.

5 Prototype System and Performance

We developed a prototype system and have evaluated its performance. Our prototype system consists of following:

- A TED device with a built-in TPM.
- A TED host PC, we used Intel Core 2 Duo 6400 with dual processors of 2.13 GHz both with 1.99 GB of RAM, with a browser (both IE 7 and 8 and Firefox 3.x tested) on windows XP OS environment.
- Our remote server machine runs a latest Tomcat web server (version 6.0) and the bank application developed using JSP. The server machine configuration is identical as TED host PC described above.
- A privacy CA developed as a java implementation. We use Java Crytographic Engine (JCE) and Bouncy Castle Crypto API to implement privacy CA. The privacy CA application is running on the server.

To measure the performance, we first tested the login without exchange of a user certificate. We then tested the login with the user certificate created and verified using the protocol steps depicted in Section 4.2. The results are summarised in Table 2.

We use industry strength RSA 2048 bit and AES 256 bit with CBC attributes for public and symmetric key encryption respectively. SHA-1 with 160-bit is used as a hash function. We use random key generation from TPM chip function. It takes

Table 2. Performance comparison results with/without user certificate

	Avg Time (s)	[Min, Max](s)
Plain UserID/Password	0.015	[0.01, 0.02]
UserID/Password with User Certificate	7.628	[7.48,7.84]

average 7.628 second using TED as an authentication tool compared only 0.015 second took using plain userID/password combination. The overhead of TED comes largely from the following protocol steps: TED initialising its resource to collect required data set and to create a cryptographic key AIK (took average 2.39 second) , creating a certificate at privacy CA (took average 1.96 seconds), and activate it after verifying the environment the key has created has not changed (took average 3.27). Currently, we are working on improving this overhead.

6 Conclusion

We have proposed a system that provides a secure environment to run user authentication in a manner that is resilient from the physical and local attacks. Our system leverages the USB-based trusted device technology such as Trusted Extension Device (TED). TED stores mission critical cryptographic keys inside the temper-resistant TPM chip. TED can be plugged into any un-trusted client machine. The protocol runs as part of our system to ensure that the userID/password is sent securely to the application server; a digital certificate for the TED user is created by a legitimated TED user; its validity is checked by a trusted third party privacy CA; the TED user's digital certificate is sent to the application server securely to ensure that the userID/password was entered by the legitimate TED user.

Unlike existing solutions in USB-based authentication token, our system utilizes standard HTML tags and JavaScript to enable communication amongst parties that are interacting. Using such standard web technology ensures our system does not require to install a special software to assist in communication on top of many heterogeneous environments. In the future, we want to verify our protocol using a popular security verification tool such as ProVerif [19]. We also plan to run a verification tool such as Fs2pv [5] for the implementation to make sure that the implementation performs in accordance with the concept we have verified using the ProVerif.

References

1. Aladdin eToken, http://www.aladdin.com/etoken
2. Barth, A., Jackson, C., Mitchell, J.: Securing frame communication in browsers. Communications of the ACM 52(6), 83–91 (2009)
3. Federal Financial Institutions Examination Council (FFIEC): Authentication in an internet banking environment, http://federalreserve.gov/boarddocs/srletters/2005/SR0519a1.pdf
4. Frischat, S.: The next generation of USB security tokens. Card Technology Today 20(6), 10–11 (2008)
5. Fs2pv: A cryptographic-Protocol Verifier for F#, http://research.microsoft.com/en-us/downloads/d54de3ef-085e-47f0-b7dc-8d56c858aba2/default.aspx
6. F-Secure virus descriptions: Cabir, http://www.f-secure.com/v-descs/cabir.shtml

7. Redbrowser, A.: F-Secure Trojan information pages, http://www.f-secure.com/v-descs/redbrowser_a.shtml
8. Gratzer, V., Naccache, D.: Trust on a Nationwide Scale. IEEE Security and Privacy 5(5), 69–71 (2007)
9. Hiltgen, A., Kramp, T., Weigold, T.: Secure Internet Banking Authentication. IEEE Security and Privacy 4(2), 21–29 (2006)
10. IronKey, https://www.ironkey.com/
11. Jackson, C., Wang, H.: Subspace: Secure Cross-Domain Communication for Web Mashups. In: 16th International Conference on World Wide Web (WWW 2007), pp. 611–620 (2007)
12. Jang, J., Nepal, S., Zic, J.: Establishing a Trust Relationship in Cooperative Information Systems. In: Meersman, R., Tari, Z. (eds.) OTM 2006. LNCS, vol. 4275, pp. 426–443. Springer, Heidelberg (2006)
13. Kolodgy, C.J.: Identity management in a virtual world. IDC White Paper (2003)
14. Mannan, M., van Oorschot, P.: Using a personal device to strengthen password authentication from an untrusted computer. In: Dietrich, S., Dhamija, R. (eds.) FC 2006 and USEC 2006. LNCS, vol. 4886, pp. 88–103. Springer, Heidelberg (2007)
15. Marchesini, J., Smith, S.W., Zhao, M.: KeyJacking: The Surprising Insecurity of Client-Side SSL. Computers and Security 24(2), 109–123 (2005)
16. Moreland, D., Nepal, S., Hwang, H., Zic, J.: A snapshot of trusted personal devices applicable to transaction processing. Jnl. of Personal and Ubiquitous Computing (2009), doi:10.1007/s00779-009-0235-6
17. Nepal, S., Zic, J., Hwang, H., Moreland, D.: Trust Extension Device: Providing Mobility and Portability of Trust in Cooperative Information Systems. In: Meersman, R., Tari, Z. (eds.) CoopIS 2006. LNCS, vol. 4803, pp. 253–271. Springer, Heidelberg (2007)
18. Parno, B., Kuo, C., Perrig, A.: Phoolproof phishing prevention. In: Di Crescenzo, G., Rubin, A.D. (eds.) FC 2006. LNCS, vol. 4107, pp. 1–19. Springer, Heidelberg (2006)
19. ProVerif: Cryptographic Protocol Verifier in Formal Model, http://www.proverif.ens.fr/
20. Shelfer, K., Procaccion, J.: Smart Card Evolution. Communications of the ACM 45(7), 83–88 (2002)
21. Trusted Computing Group, http://www.trustedcomputinggroup.org
22. Trusted Platform Module (TPM) Working Group, http://www.trustedcomputinggroup.org/groups/tpm