Trusted computing enhanced user authentication with OpenID and trustworthy user interface

Andreas Leicher* and Andreas U. Schmidt

Novalyst IT AG,
Robert-Bosch-Strasse 38, 60439 Karben, Germany
Fax: +49-6039-9154-1601
E-mail: andreas.leicher@novalyst.de
E-mail: andreas.schmidt@novalyst.de
*Corresponding author

Yogendra Shah and Inhyok Cha

InterDigital Communications LLC,
781 Third Avenue, King of Prussia, PA 19406, USA
E-mail: Yogendra.Shah@InterDigital.com
E-mail: inhyok.cha@interdigital.com

Abstract: Trusted computing, used as a security technology, can establish trust between multiple parties. One implementation of trusted computing technology standardised by the Trusted Computing Group is the trusted platform module (TPM). We build on the security provided by the TPM to create a trusted variant of identity management systems based on the popular OpenID protocol. We show that it is feasible to bind OpenID identities to the trustworthiness of the device. Our concept and implementation builds on previous work which showed that trusted computing can be used to create tickets. In this work, we use such tickets as a building block to establish trust in the OpenID protocol between the identity provider and the device. Furthermore, we investigate how mutual trust can be established in the communication between device and user during authentication. The concept of trust visualisation via a trusted environment and binding to user authentication are presented.

Keywords: trusted computing; TC; trusted platform module; TPM; OpenID; identity management; IdM; authentication; trusted token; visual attestation.

Reference to this paper should be made as follows: Leicher, A., Schmidt, A.U., Shah, Y. and Cha, I. (xxxx) 'Trusted computing enhanced user authentication with OpenID and trustworthy user interface', *Int. J. Internet Technology and Secured Transactions*, Vol. X, No. Y, pp.000–000.

Biographical notes: Andreas Leicher received his Diploma in Computer Science at Frankfurt University in 2009. With his main focus on IT security, he developed a framework for trusted computing applications, implementing a trusted ticket system by enhancing the Kerberos authentication protocol. He is working as Consultant and Senior Researcher with Novalyst IT. He is actively involved in the areas of identity management, trusted computing, TCG, IT security, privacy, mobile systems, smart card security and 3GPP/ETSI standardisation. He is author/co-author of a number of journal and conference papers and co-inventor of several US patent applications. Currently, he is working on a PhD thesis in the area of mobile identity management, especially OpenID/OAuth-based solutions.

Andreas U. Schmidt received his Doctorate in Mathematics at the University of Frankfurt/Main in 1999. After research stays in Durban and Pisa, he became Senior Researcher at the Fraunhofer Institute for Secure Information Technology, Germany, and was Area Head for security at the CREATE-NET Research Centre, Italy. He is Co-founder and Director of Novalyst IT, a consultancy specialising in IT security R&D and knowledge transfer. He produced 60+ publications in various fields, works as reviewer for renowned journals in security and served in the programme committee of numerous conferences. He organises the conference MobiSec on mobile security.

Yogendra Shah obtained his BSc and PhD in EE from The City University, London in 1982 and 1985, respectively. He has worked in the wireless industry developing consumer products incorporating wireless technologies from the early CT2 digital cordless telephony standard through to the current generation 3G systems. He is currently a Manager in the R&D Department at InterDigital with research interests in developing advanced communications modem technologies and wireless security technologies. He is an inventor or co-inventor on several US patents and has been awarded the President's and CTO innovation awards at InterDigital.

Inhyok Cha received his BS and MS from Seoul National University in 1988 and 1990, respectively, and PhD in EE from the University of Pennsylvania in 1995. He did R&D and cellular wireless R&D management for Lucent Technologies between 1996 and 2003. Since 2004, he has been with InterDigital Communications Corporation. His interest includes machine-to-machine communications, wireless communication security and trusted computing. He is the author of a number of journal and conference papers and an inventor with numerous patent awards.

1 Introduction

Trusted computing (TC) is generally regarded as a protection and security technology centred on single devices. Used as a platform-neutral security infrastructure, TC offers ways to establish trust between different entities that are separated by technical boundaries, e.g., different access technologies and access control structures. As some concepts of TC have similarities to identity management (IdM), we increase the security of a common lightweight IdM protocol for the internet, namely OpenID, by the use of TC technology. The main contribution is not only to bind the use of an OpenID identifier to a single platform, and hence provide protection from remote phishing attacks, but also to provide protection from identity theft by malware, trojans or man-in-the-browser-attacks by enabling the use of the trusted OpenID (TOID) identifier only after a successful integrity verification of the client platform.

In previous work we have presented a concept to use TC within Kerberos (Leicher et al., 2009), and have also shown that identity federation between different provider domains can be supported by TC (Fichtinger et al., 2007). This paper presents the concept and implementation of a trusted IdM protocol using the widely used OpenID protocol. Our approach of Section 2 combines attestation and integrity validation of a client system's trustworthiness with user authentication. We further demonstrate that this combination can be done efficiently in a generic demonstration environment for TC-based applications. In Section 3, a complementary technology concept to establish trust between platform and user is presented, which can close the 'trust gap' for authentication and online transactions at the user end.

1.1 TC technology

With the growing presence of computer systems in ubiquitous environments such as mobile phones, machine-to-machine communication, and sensor networks, the need for an increase in security arises. On the other hand, the enormous increase in system complexity inhibits a formal verification of the whole system. As a consequence, other means have to be established to encounter the risks and dangers to which every single system gets exposed. In a networked scenario, where multiple systems communicate, TC is a core technology to determine whether a communications partner can be trusted.

As a root of trust for the secure operation of the system, a hardware security anchor is the key to the protection of the system behaviour. Establishment of the trust boundary is conceptually associated to the boot cycle of the platform and extending trust from the root to further loaded components is a central concept of TC. This means that components that are started later on, are measured by a protected entity on the platform before they are executed, for example, digest values over a component are generated and stored protected by the root of trust. Specified by the Trusted Computing Group (TCG) this process is called authenticated boot (TCG, 2005a). Another embodiment called secure boot (TCG, 2008) adds a local enforcement engine which denies loading components whose measurement values do not match trusted reference values.

To prove trustworthiness of a system to an external party acting as a verifier, attestation mechanisms and protocols have been envisaged. They transport measurement values and verification data necessary to retrace the post-boot system state to the verifier. The trust anchor is represented by the trusted platform module (TPM) (TCG, 2007), which offers various security functions. The TPM is irremovable and uniquely bound to a particular platform. Therefore, together, the TPM and its platform form a trusted platform (TP). A more detailed overview can be found in Leicher et al. (2009), TCG (2007), Gallery (2005), and Kuntze and Schmidt (2007).

Through the TPM the TP gains a cryptographic engine and protected storage. Each TPM has a unique identity represented by an endorsement key (EK) which is created at manufacture time. The EK acts, together with the endorsement key credential (EKC), which asserts TPM conformity to TCG specifications, as a base for secure transactions. The TPM is equipped with a physical random number generator, and a key generation component which is able to create RSA key pairs. The generated private keys are kept in the secure storage protected by the TPM. Attestation protocols as defined by TCG (2005b) rest on attestation identity keys (AIKs) acting as placeholders for the EK. An AIK is a 1,024 bit RSA key whose private portion is stored securely inside the TPM. The remote attestation (RA) protocol offers pseudonymity by the use of a trusted third party, the privacy CA (PCA), which issues an AIK certificate stating that the AIK is generated by a sound TPM within a valid platform.

Instead of using a PCA to obtain AIK certificates, the TCG has also defined direct anonymous attestation (TCG, 2007, 2005b; Brickell et al., 2005, 2004; Camenisch, 2004) which uses the Camenisch-Lysyanskaya signature scheme (Camenisch and Lysynskaya, 2003) as the DAA certificate. Some research however shows that DAA cannot always fulfil the promise of increased privacy (Smyth et al., 2007; Rudolph, 2007; Leung et al., 2008). Hence, according to Camenisch (2004) and Pashalidis and Mitchell (2005), the level of privacy provided by DAA can be even lower than that offered by a PCA-based AIK certification, since verifiers in DAA get to know the identity of the issuer that attested to a platform's conformity, whereas in the PCA scheme only the PCA's identity is revealed to the verifier. The method to equalise the privacy levels proposed by Camenisch (2004) introduces additional complexity and a potential third party, which makes practical adoption of DAA questionable. DAA is not a substitute for RA, since DAA is a signature scheme which replaces the AIK certification by the PCA. Hence, DAA is another means which allows to obtain a certificate for an AIK without having to share the EK certificate with a PCA. RA however has the goal to provide a platform integrity verification of the complete system by a remote entity.

The system state is measured with the TPM acting as the central reporting authority receiving measurement values and calculating a unique representation of the state using hash values. To store the system state measurement, the TPM uses several secured registers, the platform configuration registers (PCRs). The stored measurement log (SML) keeps a more extensive record of the components and can be used together with the PCR value for the validation of the platform. Although the TCG does not specify any particular protection scheme for the SML, the AIK signed PCR value can be used to provide implicit integrity protection for the SML. This technique is based on initial work by Schneier and Kelsey (1999) for securing audit logs. An implementation of integrity measurement using the TPM is available in the integrity measurement architecture (IMA) (see Sailer et al., 2004) as a kernel extension for Linux. During an RA process the verifier receives the SML and the corresponding PCR value signed by the TPM using the AIK. The verifier can then decide if the device is in a trustworthy configuration.

As AIKs can only be used to sign data originating from the TPM an indirection must be used to sign arbitrary data. Such an indirection can be achieved by using a TPM generated signing key which is then certified by signing it with an AIK. This certified signing key (CSK) can then be used to sign arbitrary data. This scheme is shown in detail in Leicher et al. (2009) and Kuntze et al. (2006).

1.2 Related work

IdM and single sign-on (SSO) are technologies which can benefit from TC. A general analysis of how TC concepts can be applied to support SSO is discussed in Pashalidis and Mitchell (2005, 2003). The concept described in Pashalidis and Mitchell (2003) however implements the identity provider on the TP of the user, and the identity provider proves its integrity to all service providers. Such a scheme requires all service providers to keep a database of known reference values for the reported integrity measurements. Our approach is different in that we employ integrity verification of the user's platform centrally by the identity provider which in turn authenticates users and can issue statements on the platform's trustworthiness to the service providers.

In a previous paper (Leicher et al., 2009), we demonstrated how TC technology can increase the security of a ticket system, namely Kerberos, by binding the issued tickets to the client device TPM and the trustworthiness of the state of the platform to which the TPM is bound. The paper (Leicher et al., 2009) also reviews further related work with regard to secure IdM. Kerberos, however, has more architectural components which makes it less efficient than other IdM solutions. Specifically, the two interactions with authentication server (AS) and ticket granting server (TGS) add to complexity. Therefore, we looked at natural extensions of TC concepts to more compact IdM systems such as

OpenID. A specific shortcoming of trusted Kerberos is that verification of the trusted tickets must be implemented by every service provider, which requires that all of them must change their applications in a way that allows them to handle the tickets. While being feasible for small environments with few services, this concept might not be feasible for internet services which are otherwise not in a trust relationship. Furthermore, trusted Kerberos requires all participants, i.e., service providers, AS and TGS to maintain a shared key database for the encryption of the tickets. All service providers have to register with a specific TGS and share a key with this TGS to provide the service to all clients in the Kerberos realm served by the TGS. Clients that are registered in another realm cannot access this service provider unless the service provider registers with multiple Kerberos realms.

2 TOID protocol

The OpenID protocol does not specify any user authentication method to be used, and thus allows for different methods for user authentication. To claim an identity at the OpenID provider, several methods can be used, where the most common is the use of login forms, where the user provides a password. In our TOID protocol, we replace this login with a TPM-based login process. The user once registers an identity that is tightly bound to his/her specific platform (TPM). If he later decides to login using this identity, the OpenID provider challenges the platform to provide the correct credentials. In this case the credentials consist of a TPM generated ticket (credential chain). This allows the user to login without the need for a password at the OpenID provider. A local password at the user's computer can still be used to protect the identity from local attacks. The login is combined with an integrity verification of the specific platform.

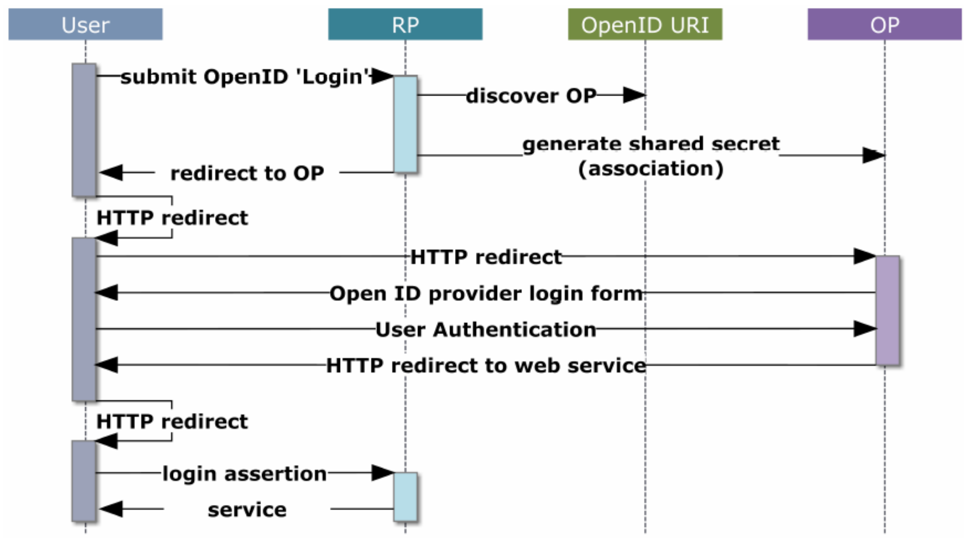
Using a TPM signed statement on the system configuration values, the OpenID provider can compare the reported system state to previously generated reference values, allowing only trustworthy clients to login and claim an identity. This combined authentication and attestation allows for a fine grained access control by not only binding the authentication data to a specific platform but also to a predefined system state which is considered trustworthy. This enables OpenID to be applied to new use cases requiring enhanced security and detection of unauthorised modifications to the system.

2.1 OpenID protocol

As an open, decentralised IdM framework, OpenID (2010a) was developed to provide a single sign on experience to users across services on the internet. With OpenID, it is possible to sign on to different services, with a single identity, called OpenID identifier. OpenID hence eliminates the need to create separate logins and passwords for the services the user wants to access. OpenID, often seen as IdM for the Web 2.0, is supported by major internet companies, including AOL, Facebook, Google, Microsoft, Yahoo, etc. The latest report of OpenID usage (OpenID, 2010b) counts over 1 billion OpenID enabled accounts and over 9 million websites utilising OpenID for registration and login. During the last year efforts were made by the OpenID Foundation (2010) and the US government to deploy OpenID on federal websites.

With OpenID (see Figure 1) users are provided with a single identity which they can use across a variety of services, without having to remember different identifiers and login information. The websites supporting OpenID login are referred to as relying parties (RP). OpenID uses identifiers in the form of an URI, e.g., http://foo.myopenid.com/, provided by OpenID providers (OP), which are used to sign in to the RPs.

Figure 1 Overview of the OpenID protocol (see online version for colours)



To sign in to the RP, the user enters his OpenID identifier in a login form. The RP reads the identifier URI and extracts the necessary information to redirect the user's browser to the OP. The OP and RP then establish an association using a shared secret, which is used to sign any future communication between the OP and the RP. After establishing the association, the RP redirects the user's browser to the OP login page. The user authenticates at the OP, using the method provided by the OP, e.g., via user-name and password. The actual authentication mechanism between user and OP is not dictated by the OpenID specifications and hence leaves space for alternative authentication methods involving smart cards or other security tokens such as the TPM. After user authentication, the OP redirects the user's browser back to the RP, including a signed assertion on whether authentication succeeded or not. The user is then logged in at the RP and can use all services provided by the RP. OpenID allows users to access services provided by the RP while not having to register with the RP directly.

Tsyrklevich and Tsyrklevich (2007) is a white-paper outlining security issues with OpenID. One of the biggest concerns when dealing with OpenID is the danger of phishing. If an attacker is able to trick users into giving away their credentials to a fake OP, owned by the attacker, he can access a broad range of services in the name of the legitimate user. Since OPs host the identifiers for multiple users they are a rewarding target for the attacker as they can collect multiple OpenID identifiers. Redirecting users to the fake OP can be accomplished by the attacker by setting up a malicious RP which redirects the user to the fake OP instead of the original one. The attacker hence does not need to attack the OP directly. Other attacks include man-in-the-middle (MITM) attacks during the association between the RP and the OP, replay attacks, involving sniffing the session identifier of an authenticated session as well as cross-site-request-forgery (CSRF) attacks which silently log the user on to other sites once they have logged in into another OpenID site and perform actions in the user's name. Such a CSRF attack mainly relies on the fact that the OP and not the RP decides on the user login security policy.

2.2 Integration of TC concepts

The integration of TC concepts into the OpenID protocol allows countering some of the threats mentioned above and leverage of the overall security of OpenID. Four entities are involved in the TOID authentication process:

1 the user accessing a service

2 the OP supporting trust validation

3 a PCA to certify AIKs from the user's TPM

4 a RP using OpenID authentication.

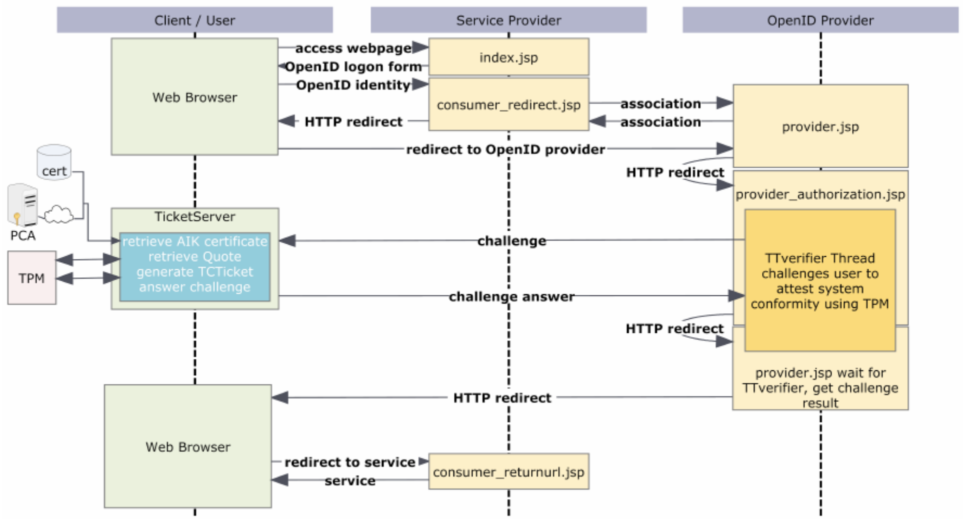
We will assume that the following requirements hold on these entities: A PCA, chosen by the user has to issue an AIK certificate, which is a X.509 certificate that includes additional private extensions defined by TCG (2005b). We use the following indirection, as described in more detail in Leicher et al. (2009) and Kuntze et al. (2006) to obtain a CSK: The TPM generates a new RSA key pair, keeping the private part secured by the TPM. After signing it with the AIK using the internal TPM operation TPM CertifyKey, this key is referred to as a CSK and can subsequently be used to sign arbitrary data. The AIK certificate provides a binding of the AIK to the platform since the PCA checks the relevant TPM certificates during the AIK certification process, i.e., the PCA states that this AIK is generated and stored securely in a sound TPM. The verifier, upon receipt of the AIK signed CSK and AIK certificate, is able to verify that the CSK is a TPM-secured key by deriving the trust from the PCA issued AIK certificate. The PCA will hold all platform specific information as included in the EK certificate and platform certificate which the PCA receives and checks during AIK certification. After certifying an AIK at the PCA, the user can choose an OP supporting the TOID protocol to host his claimed identity. In order to register such a claimed identity, he must provide a valid AIK certificate to the OP.

It is important to note that the user does not establish any shared credentials with the PCA for the AIK certification. The user generates the AIK locally inside the TPM and is able to establish a password which must be used to unlock usage of this key from the TPM. This is a local secret, which is checked by the TPM and never shared with an outside entity. The PCA is the only instance that will be able to resolve a claimed identity to a real platform, by keeping a secured database that maps EK certificates to AIKs. The RP only has to enable OpenID login for his site and has to accept assertions from this OP. Different AIKs can be used by different users, where each AIK is protected by the TPM to be used only by the user who knows the authentication secret (stored in the TPM) for that user's AIK. Hence, each user in a multi-user system environment can create an own AIK and associate his OpenID identity with this AIK in the registration process with his TOID enabled OP.

2.2.1 The trusted ticket server

We implemented both the trusted client and the enhanced OP server side of our concept in Java, based on the project OpenID4Java (2010). It is invoked from a JSP page directly and can be used to associate a service provider using OpenID login and to authenticate a user accessing the service with his OpenID. In order enable TPM-based authentication, classes and methods had to be added to handle the integrity validation and authentication with the trusted ticket server (TTS) component running on the client machine.

Figure 2 Overview of the protocol flow for TOID (see online version for colours)



The TTS runs as a service application on the user's machine and is responsible for the authentication toward a TOID OP. It is a trusted functional entity which is deployed in a trusted execution environment (TEE) on the client machine. The TEE provides an isolated, integrity protected and secured execution environment for the TTS. In general, we can assume that such a TEE is available, provided mainly by the TPM and OS functions. One approach to gain such a TEE is presented by Balfe and Paterson (2008) who implement Europay-Mastercard-Visa (EMV) functionality as a trusted application using virtualisation. Another approach, based on virtualisation is presented by Gajek et al. (2009). They use a security kernel to securely isolate a trusted component which acts as a credential store for website login information. In general, the protocol consists of the following steps, shown in Figure 2.

1 *Initial connection to the RP:* The user accesses the website of the service provider (index.jsp). If the user wants to login using his OpenID URI, the consumer redirect.jsp page at the service provider connects to the given URI and thus retrieves the address of the OP hosting the claimed identity.

2 *Association of service provider to OpenID provider:* According to the OpenID protocol, the RP associates with the OP. This includes a secure exchange of the request, the claimed identity and a return URL to which the client will be redirected by the OP if authentication is successful. These steps are performed on the server provider side using consumer redirect.jsp and on the OpenID provider side using

provider.jsp. After the association is established, the client is redirected to the webpage of the OP. The page checks if the user is already logged in, and if not redirects the user's browser to the OP login page provider authorization.jsp using HTTP redirect. The redirection address is retrieved from the user supplied identifier.

3 *Authentication of the client:* The provider authorization.jsp page of the OP, requests authentication and user authorisation to log in to the RP. After clicking a link on the page, a new background thread starts which challenges the TTS on the client side. The provider authorization.jsp redirects the user back to the provider.jsp page, which waits for the thread to finish and evaluates the result of the challenge.

4 *Redirection to the service provider:* The OP (provider.jsp) redirects the user to the consumer returnurl.jsp page at the service provider. The consumer returnurl.jsp checks that the redirect comes from the associated OP and grants access to the user.

In our implementation, the TTS must be trusted to handle AIK certificates and CSKs properly, protect the creation process of the tickets and collect and report platform validation data to the OP. The TTS listens on a predefined port and waits for incoming challenges. Upon receipt of a challenge, containing the identity the user wanted to use in OpenID and the service request he issued at the service provider, the user is required to explicitly allow the challenge. Every challenge is shown to the user along with the transaction details in the TTS screen. In the current proof of concept implementation, the user has to visually compare the challenges shown in the browser window and in the TTS screen. This separate, secure user interface (UI) can protect the user from input phishing on the client side, e.g., using the rudimentary feature of a colour-coding for the secure UI as described in Gajek et al. (2007). Other solutions than a visual comparison could be used, e.g., that after successful integrity verification the OP calculates a session nonce which is then cryptographically bound to the integrity state as reported by the TTS, using the TPM sealing functions. The TTS then decrypts the nonce and displays it to the user who has to provide it as a one time password in the OP authentication page. In principle it is possible to implement additional protection methods, such as OP authentication performed by the TTS, which could be done, e.g., by the use of TLS/SSL certificates. Additional measures to prevent denial of service (DOS) attacks against the TTS could be implemented, e.g., by integrating the TTS functionality as a trusted browser extension, similar to the work of Bal et al. (2009), which is able to communicate with the browser such that the TTS would only accept incoming requests if an OpenID authentication session is currently taking place. Depending on the desired level of security, such a trusted browser extension could also store the user decision per session, to further increase the single-sign-on experience. If the challenge is accepted, the user is prompted to enter the password for the AIK corresponding to the given identity and to authenticate for TPM usage by giving the SRK password. The TTS then tries to retrieve a previously acquired certificate for this identity from the local certificate storage. If no certificate can be found in the local database, the user can choose a PCA to connect to in order to undergo an AIK certification process and obtain a certificate for the AIK. Therefore, he must supply the correct owner password of the TPM. This prevents creation of rogue identities by other persons than the owner of the TPM. Especially in corporate environments, where the owner of the TPM is often not the user but another entity, this allows the administrator to control the enrolment process for new OpenID identifiers in the corporate network. The TTS receives a random nonce from the challenger. An

AIK-signed quote, including the nonce is retrieved from the TPM, providing a statement about the system's state. Next, the TTS creates a signed ticket which involves the creation of a CSK that can be used to sign the request and the identity. The information needed to verify the signatures is included in the ticket, so that the receiving party can easily verify the ticket. Together with the SML the ticket is sent back to the challenger. Figure 3 shows the authentication flow between the TTS and the OP.

Figure 3 Detailed protocol flow for TTS-OP communication (see online version for colours)

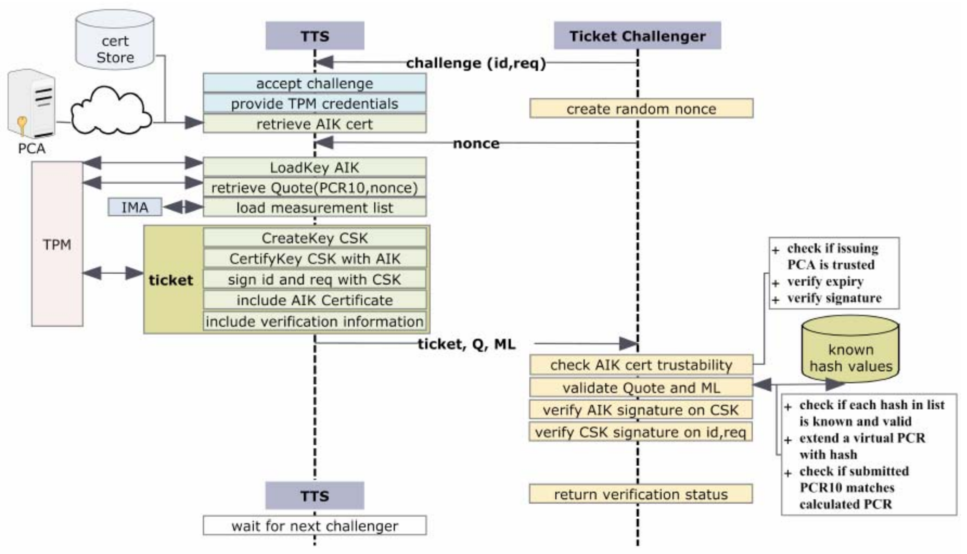
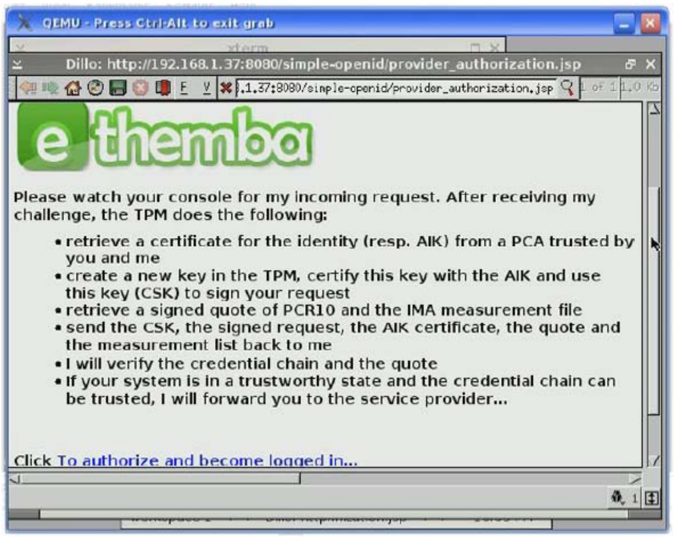


Figure 4 Screenshot showing the OP login page in the browser (see online version for colours)

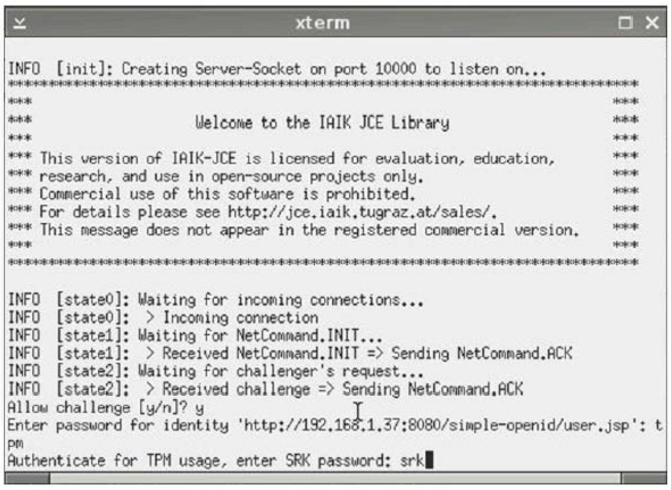

QEMU - Press Ctrl-Alt to exit grab
Dillo: http://192.168.1.37:8080/simple-openid/provider_authorization.jsp
.1.37:8080/simple-openid/provider_authorization.jsp
e themba

Please watch your console for my incoming request. After receiving my challenge, the TPM does the following:

- retrieve a certificate for the identity (resp. AIK) from a PCA trusted by you and me
- create a new key in the TPM, certify this key with the AIK and use this key (CSK) to sign your request
- retrieve a signed quote of PCR10 and the IMA measurement file
- send the CSK, the signed request, the AIK certificate, the quote and the measurement list back to me
- I will verify the credential chain and the quote
- If your system is in a trustworthy state and the credential chain can be trusted, I will forward you to the service provider...

Click To authorize and become logged in...

Figure 5 Screenshot showing the TTS after receiving the challenge from the OP prompting the user for the local passwords



```
INFO  [init]: Creating Server-Socket on port 10000 to listen on...
***************************************************************************
***                                                                     ***
***                  Welcome to the IAIK JCE Library                    ***
***                                                                     ***
*** This version of IAIK-JCE is licensed for evaluation, education,     ***
*** research, and use in open-source projects only.                     ***
*** Commercial use of this software is prohibited.                      ***
*** For details please see http://jce.iaik.tugraz.at/sales/.            ***
*** This message does not appear in the registered commercial version.  ***
***                                                                     ***
***************************************************************************

INFO  [state0]: Waiting for incoming connections...
INFO  [state0]:  > Incoming connection
INFO  [state1]: Waiting for NetCommand.INIT...
INFO  [state1]:  > Received NetCommand.INIT => Sending NetCommand.ACK
INFO  [state2]: Waiting for challenger's request...
INFO  [state2]:  > Received challenge => Sending NetCommand.ACK
Allow challenge [y/n]? y
Enter password for identity 'http://192.168.1.37:8080/simple-openid/user.jsp': t
pm
Authenticate for TPM usage, enter SRK password: srk
```

In order to develop and implement our concepts for TOID, we set up a trusted demonstration environment called ethemba (Leicher and Brett, 2010). The goal was to design a system in which it is possible, without the need of a physical TPM, to access all desired TPM functions. We used the TPM emulator of Strasser (2004) as base and to simulate a complete system we established a connection between the TPM emulator and QEMU, a virtualisation emulation environment, enabling our virtual machines to execute TPM applications. Our virtual machine uses a standard debian Linux distribution, with a patched kernel to support IMA (Sailer et al., 2004), enabling measurement and logging of every component the kernel loads. The ethemba framework also includes a PCA implementation and support for the TCG RA protocols. It is partially based on jTSS (trustedjava, 2010) and has already been used in various other applications (Leicher et al., 2009; Bal et al., 2009; Brett et al., 2009). For the TOID application we extended it with the implementation of all necessary OpenID functions on both client and server sides. Figures 4 and 5 show screen-shots of our implementations of the OP and the TTS.

2.3 Device authentication and validation by the OP

During the TOID protocol, the OP receives the following information from the TTS:

1 the signed quote from the TPM, including the nonce as anti-replay protection

2 the plain-text measurement file

3 the ticket, including the signed identity and request string, the public key portion of the CSK, the AIK signature on the CSK and the AIK certificate issued by the PCA.

In order to authenticate the client, the challenger first checks the validity of the AIK certificate. This validation includes the verification of the PCA signature on the AIK certificate. The challenger then has to verify the ticket, i.e., the credential chain incorporated into it. Therefore, the AIK signature on the CSK public key hash in the ticket and the CSK signature on the service request and identity in the ticket are validated. To validate the trustworthiness of the system, the challenger then checks the entries in the received SML against a database of known good values. This step is accompanied by recalculating the expected PCR value which is, in the final step, compared to the reported signed PCR value. If at any step in this process verification fails, the client will not be authenticated. In this protocol, the OP receives the platform configuration in order to verify the platform integrity. Therefore the OP must be trusted by the user not to reveal this configuration to unauthorised third parties. Since the user in OpenID already trusts the OP not to misuse his OpenID identifier, this is an additional requirement on an entity the user already has an established trust relationship with. If the RP should however get some information about the authentication mechanism, e.g., TPM-based vs. non-TPM-based or additional assurance of the platform integrity, additional mechanisms can apply. The OP might be offering this authentication as additional service and guarantees integrity checking for RPs by contractual agreements, and hence does explicitly disclose a platform's properties to any RP. In security demanding applications however, it might be desirable to signal the outcome of the integrity verification process to RPs, which can be accomplished by including this information in the OP-signed assertion message to the RP. Such information could for example include details on the platform, the type of integrity checks performed, or along the lines of a property-based attestation (PBA) according to Sadeghi and Stüble (2004), by reporting the platform conformance to a certain set of properties.

TOID achieves user authentication and device trust attestation at the same time by the following procedures: user authentication is achieved because the user must already have pre-registered the certificates for the AIK and/or the CSK with the OP, and the device (or more precisely, the TPM that is hard-bound to the device), will send data that is signed with the private key of the AIKs and the CSK in the step of challenge response. Therefore, the OP, upon verification of the public portion of the keys, and upon verification of the fact that the user is using the same AIK that had been pre-registered, will know that the user who used the device to send the challenge response is the same user who had been pre-registered with it. Device trust attestation is of course achieved because the OP can verify if the TPM quote of the PCR values and the measurement logs, and the part of the ticket (identifier and request) are all signed using the now verified AIK/CSK. If both match, then the OP verifies that the client used to send the OpenID request is in fact trustworthy. If only the verification of the user and the request is achieved but the comparison of the PCR values and the measurement logs fail, then the OP knows that the trust-related data now indicates that the device may be compromised and is in a different configuration state than expected.

2.4 Analysis of TOID

Our solution does not require any changes to the OpenID protocol standards. The OpenID specifications do not foresee a single method for user authentication. We therefore developed an enhanced authentication concept which basically would allow OPs to differentiate among themselves, with one function being the assertion of enhanced security features. Such assertions are interesting for the users since they can be assured, that their OpenID identity is well protected, especially if it is HW-bound and even more important, such OPs enable RPs to rely on the information received from them, e.g., enabling banks, government and other security-demanding services to use OpenID with a white-list of 'security-aware and certified' OPs.

Some of the mentioned vulnerabilities of the standard OpenID protocol are addressed by the OpenID security best practises (OpenID, 2010c). Phishing however remains one of the best known attacks and is a main problem for the OpenID protocol. Since the identifier is used with all RPs, the security of this identifier should be of special concern. Replacing multiple insecure passwords or a single password which is used across multiple RPs with a strongly authenticated identity reduces the spread of secret data, which is especially important if the same password is used across all RPs. Recent attacks, gathering credentials from social networks have shown that proliferation of credentials is an important security issue. Centralisation of credentials at a trustworthy OP also allows the user to control the spread of personal information in a privacy protecting way. With the use of a single point of authentication, namely the OP, it is easier (and cheaper) to build strong (e.g., multi-factor) authentication between the user and the OP, which then translates to a strong authentication between the user and all RPs that he uses. We follow this direction with the TPM providing a secure storage for the credentials, paired with a local authentication where no credentials are released directly to the OP, and binding the authentication to the platform the user initially registered the identifier with. The credentials are bound to the platform's TPM and stored in the secure storage of the TPM. No password or user name is sent to the OP in our approach, which in turn renders phishing attacks as described in Section 2.1 on the OP side useless. The phishing attacker's target is to steal the user's credentials by setting up a fake OP at a different website controlled by the attacker and then re-use the login credentials with the real OP later on. In TOID, an attacker would not be able to retrieve re-usable credentials with a fake OP. The attacker's fake OP is assumed to use another website than the legitimate OP, and hence cannot impersonate the legitimate user at a RP even if the user is tricked into approving a challenge from the fake OP, since the user's OpenID identifier is hosted at the web address specified in the OpenID identifier, which is the real OP. Hence, the attacker would have to perform an online attack in which he is able to trick the user into approving the challenge and at the same time control the URL of the real OP, to be able to establish associations between his fake OP and arbitrary RPs. Such an online attack is possible with the normal OpenID protocol as well. The attacker is required to control the OP to RP communication channel as well as the OP to user communication channel at the same time, resulting in a very limited and targeted attack scenario. TOID can increase the security even in this scenario if security demanding RPs are requesting signed platform integrity information from the OP which cannot be provided by the fake OP. Replaying fake challenge responses to the real OP is not possible since the OpenID protocol uses a nonce for every session. Faking challenge responses would require the attacker to gain access to the TPM protected AIK and CSK respectively for the creation of the response signature. By the inclusion of a replay protection inside the ticket, the problem of a phishing OP is mitigated. Additionally, a MITM attack, where the attacker is able to sit between the TTS and OP, can be prevented by a mutual authentication between OP and TTS, in which the TTS securely stores and verifies the OP certificates. To support multiuser environments we are able to establish multiple identities, represented by AIKs stored in the TPM, on a single platform and are able to protect them with a local password, secured by the TPM. Each user creates a new unique AIK and performs the AIK certification protocol. AIK certification is integrated in the AIK creation process, can be done before running an OpenID authentication session and is needed only once per AIK. By the additional inclusion of platform integrity validation by the OP, we further increase the security by detecting malware or otherwise modified systems and preventing them from using the OpenID identifier.

Compared to simple TLS authentication with a TPM-protected key, TOpenID combines attestation and authentication, whereas TLS alone only provides authentication. Using TLS extensions with integrity attestation is an option to integrate platform information. However, using TLS every RP will have to support the verification of the TLS credentials as well as the integrity verification of the reported measurement values which is highly impractical. Hence using an established and widely adopted lightweight IdM solution such as OpenID and bridging it with additional security and trust information bears an increased benefit, since a secure web-SSO can be provided to a large variety of RPs without any modifications to the existing OpenID authentication mechanisms implemented at the RPs. The presented scheme binds the use of an OpenID identifier to a specific TPM and hence a platform and the platform's integrity. By this binding an increased level of security can be achieved, as well as the use of OpenID for security demanding applications could be enabled. The goal of the current contribution is to enhance the security and bind identity authentication to a single platform. This effectively blocks attackers from transferring authentication credentials to a different machine, as it is done it typical phishing attacks. This contribution does not yet address the issue of identifier migration, e.g., using the same identifier with different devices, such as a laptop, smart-phone and PC. Several mechanisms for migration are discussed in the context of virtualisation architectures (Plaquin et al., 2009) and could be applied to the presented solution.

3 A concept for trustworthy UIs

The basic TOpenID protocol derives its increased security from two key factors: validation of the user device by the RP using RA performed by the TTS, and binding of the OpenID authentication proper to the attestation via the AIK protected by the TPM. This closes the trust gap between device and RP, but not between device and user. In fact, trojan's, malware, and phishing attacks pose serious risks to users performing transactions with remote parties on their devices (see Dhamija et al., 2006). More sophisticated attack vectors are realised, according to Jackson et al. (2007b), by so-called transaction generators, which produce and execute fraudulent online transactions to the profit of the attacker. A requirement for a trustworthy authentication and online transaction ecosystem is the effective mitigation of MITM and man-in-the-browser attacks. A platform-centric requirement elucidating the necessary chain of trust over multiple nodes, may be formulated as follows:

> The platform is in a trusted state to run a trusted application to act on correct data to perform the desired transaction with a trusted remote party.

From this derives the specific requirement to close the trust gap at the user end:

> *Visual attestation (VA):* ability of a platform to validate an application and according data to a human validator, attesting to correctness of a transaction, using only human perception.

At the heart of the problem lie limitations to platform attestation. In practise, hardly any user platform can be considered a TCB, i.e., an unconditionally trusted user device. Such a device would not need any special provisions for VA, but its trustworthiness would be identical with its operational state. But for 'standard' devices a complete attestation of the total software/hardware configuration is not practical. Concepts of separation into trusted and untrusted domains can be used to validate the components needed to meet the above requirements in a concrete context, for instance those which are necessary for secure operation of TOpenID. See Schmidt et al. (2010) for an extensive discussion of the concepts around remote platform validation and management. We believe that every system for user authentication and trustworthy online transactions must take this main limitation of real systems into account.

In the concrete scenario of user authentication with TOpenID, if only parts of the device are validated by the IdP, it must be ensured that the user confirms the use of the OpenID with the TTS and a trusted browser, and that this input cannot (easily) be intercepted or replayed in the manner of a MITM attack. Hence, the trusted device domain needs a trusted UI which protects the input and at the same time indicates to the user that he is using his device in a secure mode, i.e., it performs VA. The simplest form of such an indicator would be a LED which can only be lit if the device (processor) is run in trusted mode. However, such a simple approach does not respect the requirement to bind VA to a specific transaction, e.g., a TOpenID authentication and login.

From the user's perspective, i.e., classifying by the form of user involvement, there are three basic modes of VA.

V1a User notification by the platform using uniform visual cues to symbolise trustworthiness.

V1b User notification with a visualisation of a secret, e.g., an image or an alphanumeric value, shared with the user and endowed with additional security properties, e.g., binding to a specific platform, time-dependency, inclusion of data specific to a transaction.

V2 Challenge-response between user (challenger) and platform, in which the platform gives a correct answer to the user challenge using some secret.

Beside the mentioned indicator LED, an example for options V1, although for attestation of a remote platform, is represented by extended validation certificates displayed by modern browsers to indicate trustworthiness of a remote website (Jackson et al., 2007a). A generic method to achieve VA in case, which involve a secret, is to bind this secret to the trustworthy state of the platform (in the case of TC called sealing as described above for the TTS secrets). A classic prototype for option V2 are press-to-test buttons found on some programmable calculators (Rosenquist, 2009).

Since V1a cannot be specific to a transaction and platform (state) we will concentrate on the modes 1b and 2 involving a secret. It is straightforward to formulate requirements for a secure UI supporting VA for the latter two modes (with some overlap between requirements).

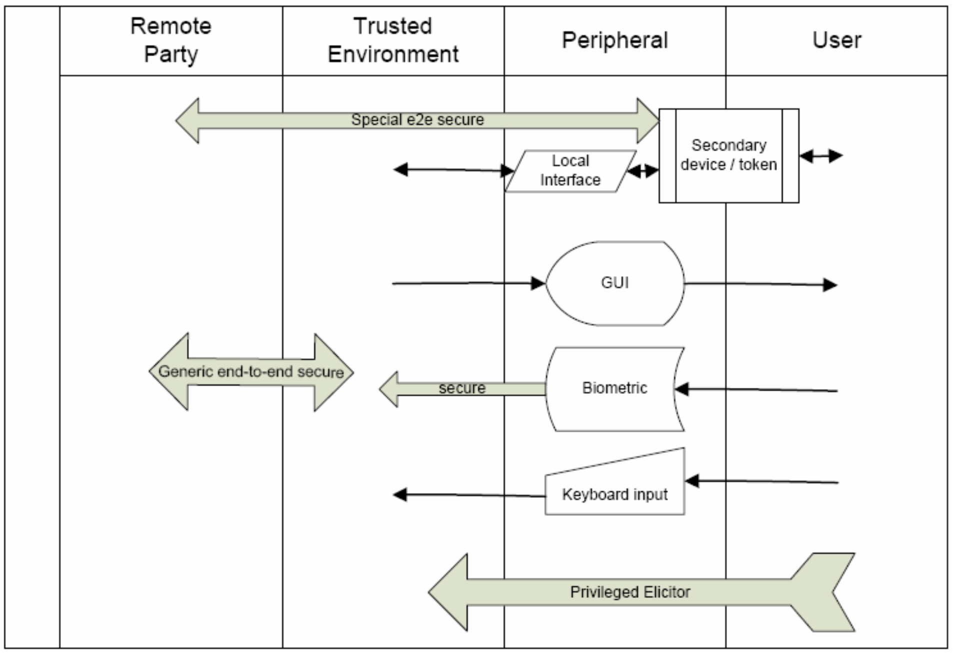
- the secret is hard to (capture and) replay
- the secret is hard to re-produce by a malicious application
- it is difficult to intercept and suppress the display of the secret (to allow an attacker to show another fake secret to trick the user)
- it is easily possible for the user to distinguish the real secret from a fake one

- it is hard to compute the secret given its visual representation, i.e., near-one-way property of the visualisation
- it is difficult for the user to neglect, or bypass, VA.

The last item warrants some comment. The study Schechter et al. (2007) concluded that users tend to ignore VA (or worse, absence of it) on websites, even in cases where this posed high risk, for instance in online banking. This speaks for requiring some interaction from the user but concurrently keep VA as unobtrusive as possible. This shows how VA is a textbook example for the difficulty to reconcile security and user experience. The visualisation part of VA, i.e., establishing a secure UI and visualisation of a secret proper, is a topic o ongoing research. One appealing idea is that of 'visual hashes' by Perrig and Song (1999). They propose an image representation of numerical values satisfying near-one-way properties, i.e., collision-freeness and pre-image resistance, measured in terms of indistinguishability by humans.

Whatever visualisation method and properties of a secret are used for VA, establishing a functional environment for it that allows practical deployment, and combines usability with security, remains a pre-condition. In the following, we describe a concrete proposal for such a system. The aim is to construct a partially secure UI which realises option V2 (challenge-response) in a very simple form, necessitating only partial security of the underlying platform and is procedurally bound to a transaction, here concretely user authentication. We call this combination of functionalities a *trusted visual token* (*TVT*).

Figure 6 Security classification of I/O paths from a system secure environment to user and communication path to a remote party



3.1 A privileged elicitor and TVT

A key ingredient to the realisation of the user challenge for the TVT is a special input path from user to a secure environment on the platform in which the TVT functions reside. Considering the generic I/O paths from user to a trusted environment (TrE) or a secure element on the platform, and from there to a remote party, the coarse classification of Figure 6 may be applied.

At the top, a secondary device communicates via a local link with a TrE to establish a secondary channel. In turn, the secondary device may have a secure, secondary channel with the remote party which can be used for secure transactions. Typical examples are codes sent to a user's cell-phone via SMS, which are then either manually input into some client application (e.g., a transaction number for online banking), or, in modern versions such as split-terminal authentication proposed by 3GPP (2010), transferred via a local wireless link. This complex form of interaction represents the current state-of-the art of user authentication and transaction authorisation, but it is complex. On the second row, a genuine UI is very difficult and costly to secure on arbitrary devices. Third, biometric input is very secure but it is a general purpose method for user authentication, rather than for general input. Also, it is either too obtrusive (swiping a finger) for the simple challenge required for V2, or it is passive (e.g., a picture of the user is taken by the platform's web cam) and does not allow to pose an own-controlled challenge to the platform. Fourth, manual input via a keyboard is easy to attack, in view of the now commonplace attacks on ATMs using fake keyboards, or key logging malware on PCs.

Here, we propose the simple idea of a *privileged elicitor* (*PE*) (borrowing a term from biology) as a dedicated input path for posing a challenge to establish V2. The PE is a secure input path (for instance, a key) for the user to his platform, where the endpoint inside the platform is a TrE or secure element, i.e., some execution space which is trustworthy for the purpose of the desired transaction. The PE represents a binary signal to the TrE that activates the TVT for performing VA. The combination of PE and visual attestation in the TVT requires specific security features which may be analysed on the three traditional axes of security, applied to the binary ('ON/OFF') signal it represents.

- *Integrity* means protection against false ON signals of the PE. But since the user would notice false events, i.e., a VA occurring without the user having triggered the PE, the threat of false ON is practically ruled out.
- *Availability* means protection against false negatives, i.e., suppression of the ON signal of the PE by the attacker. Again, since the user would notice false negatives in the challenge-response procedure that links PE to VA in the TVT, this threat is mitigated.
- A threat to *confidentiality* means for the PE-TVT combination that an attacker would be able to divert the PE's ON signal to an untrustworthy endpoint, and produce a fake visual attestation to the user. The threshold for this attack is very high since the attacker must break integrity and availability as described before, and furthermore he must be able to obtain secrets and current data from the original platform's TrE to reproduce the VA expected by the user. The ancillary condition that the attacker has to do all this without noticeable delay, further raises the threshold for this attack.

We are now in a position to give a functional description of TVT. the TVT is a trustworthy entity on the user's platform, for instance realised in a software TrE, or in a hardware secure execution environment (such as a smart card), which responds in a pre-defined manner with VA to the user challenge posed via the PE. It has the characteristic features:

- VA proving the TVT's trustworthy state to the user, is displayed to the user on the platform's main or another, dedicated display
- TVT uses hardware-secured (smart-card, TPM-sealed, etc.) secrets for VA
- TVT has access to methods for authenticating the user, e.g., biometric input
- the TVT's trustworthy state can be validated by a remote party, e.g., using RA
- the TVT may validate other components of the platform, for instance a browser or an on-line banking application, and incorporate information about those components' trustworthiness in VA
- TVT has access to data specific to a particular transaction, and is able to incorporate such data in a meaningful and unique way in VA.

The above features may work in combination as required by the specific purpose. Not all of them may be needed in every case.

3.2 TVT operation and architecture example

A basic usage of the TVT in a transaction is that of user authentication, either locally or to a remote party. This process, here generically called 'log on', works as follows. When the user wishes to log on to the platform or a remote service, he opens a log-on application, or browses to a log-on page of the service on the web. Then, the user presses the PE and obtains VA. The VA may look as shown in Figure 7, combining several features such as a secret (time-dependent) image, user notification information, platform information, freshness information (current date/time). The TVT architecture depicted in Figure 8 shows the particular use case of log-on to a single-sign-on domain, where the remote party ultimately receiving the user authentication is a domain controller of a network domain (acronyms used are listed in Table 1.

Figure 7 Example TVT log-on screen combining security features (see online version for colours)

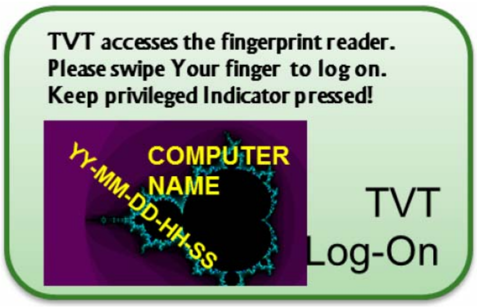

TVT accesses the fingerprint reader.
Please swipe Your finger to log on.
Keep privileged Indicator pressed!

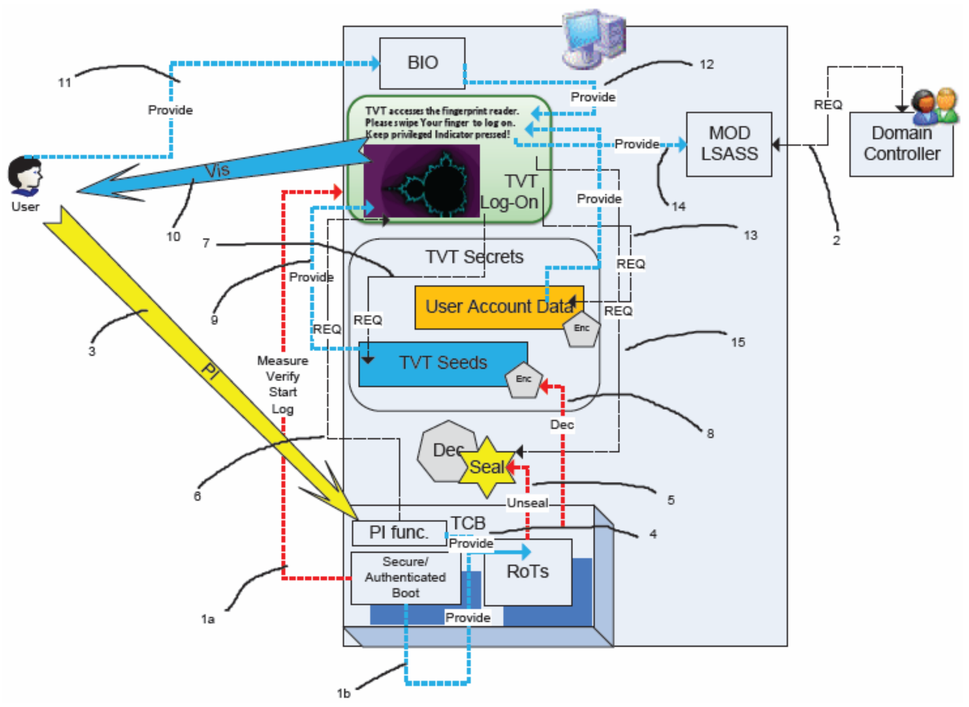
Table 1 Acronyms used in TVT architecture and log-on scenario

MOD LSASS	Modified (to receive attestation data) local security authority subsystem (a Microsoft WindowsTMcomponent responsible for local and network user log-on
ENC	Keys to encrypt TVT secret data
DEC	Decryption keys for ENC
REQ	Data request
RoT	Root of trust, e.g., a hardware secure element such as a TPM
BIO	Biometric authentication device

In a secure or authenticated boot process at system start-up, the platform's RoTs measure and store the state of the TVT application (1a), for instance in the platforms' PCRs (1b). Optionally, the TVT application may be contained in a secure environment which protects s the TVT also at run-time. The measurement facility, as well as the RoTs are contained in the TCB of the platform. When a user wants to log on to a network domain using this platform, the domain controller requests user credentials from the platform (2). TVT activation occurs at this point and the remotely requested TVT user logon is initiated, that is, the remote service signals to the TVT on the platform that it requires user authentication (preferably via a mutually authenticated channel), upon which the TVT displays a generic log-on notification, requiring the user to trigger the PE (3). The PE signal is transmitted to some PE functionality contained in the TCB. The PE functionality unseals the TVT master decryption keys (4). That means that PE functionality issues an unsealing request for DEC to the platform RoTs, quoting the state of TVT. RoTs check the state and decrypt DEC (5), if the state is correct (unsealing). The DEC keys are keys which are part of a key hierarchy which can only be used inside the TCB, for instance keys of a TPM key hierarchy. DEC can from this point on be used to decrypt the TVT VA seeds which are encrypted with corresponding encryption keys ENC. The PE functionality commands the TVT application to visually attest to the user (6). The TVT application requests decryption of the TVT seeds (7), which is performed using DEC (8), and the TVT seeds are provided to the TVT application (9). Using them, the TVT application performs VA (10), including the request for local user authentication, e.g., using BIO. The user authenticates (11), and the BIO facility signals authentication success to TVT (12). (BIO may be assumed to be part of the TCB or otherwise sufficiently secure.) TVT requests user credentials from its user account data storage, e.g., a user name and password to be used for network log-on (13). That data is also decrypted and sent to TVT which in turn provides it to the log-on application LSASS (14), which in turn forwards it to the network domain controller. As a variant, DEC may remain sealed and is only unsealed on-the-fly when TVT issues a request for any secret (15).

The TVT is equipped with two secured data stores which serve to make it a multi-purpose tool for VA and secure transactions. TVT seeds contain all secrets that TVT uses to visualise anything to the user and to communicate securely with remote entities. That comprises seeds for visualisation [e.g., numeric values which can be visualised by methods as described by Perrig and Song (1999)], TVT Individual secrets (specific to a platform), TVT credentials for secure communication with other entities, user defined (security) parameters, perapplication secrets, user enrolled secrets such as private images. On the other hand, user account data enable the functions of the TVT which are similar to a password vault. They comprise, but are not limited to biometric user reference data, domain, remote service and user name lists and associations, user credentials, e.g., passwords or hash values of passwords, credentials for authentication of a remote service or domain controller (e.g., digital certificates). The described construction and operation of the TVT shows that it can be a flexible instrument for VA, user authentication and general transaction authorisation.

Figure 8 TVT architecture and log-on procedure (see online version for colours)



4 Conclusions

OpenID is a lightweight protocol for federated IdM and is rapidly being adopted by the industry. We have shown that by incorporating TC technology into the protocol, it is possible to create a more robust and secure OpenID implementation, which subsequently enables the protocol to be used for secure transactions such as financial payments etc. We have shown that by binding OpenID identities to the trustworthiness of the device, assured by the device's TPM, and then relaying the trust information to the OpenID provider, we can establish trust between the OpenID provider and the device. We have also implemented our concepts on a virtualisation environment, faithfully implementing all functions required of the TPM, the device, and the OpenID provider in order to implement the TOID protocol. Many laptops and desktops now incorporate a TPM and trusted software stacks, facilitating the introduction of this improved OpenID protocol.

Complementing the proposal for TC enabled user authentication is the concept of a TVT to establish user trust in a platform during a transaction. Taken together TOpenID and TVT potentially enable security for online transactions which to date require extra hardware and a re cumbersome to users in many cases. TC enables these operations with high security, in principle, on any computing platform.

Acknowledgements

This work was funded by InterDigital, Inc. The author would like to thank Lawrence Case, Bob DiFazio, David Greiner, Louis Guccione, Dolores Howry, and Michael V. Meyerstein, for many useful discussions and comments.

References

Bal, G., Kuntze, N. and Schmidt, A.U. (2009) 'Injecting trust to cryptographic key management', *11th Intl. Conf. Advanced Communication Technology (ICACT 2009),* Phoenix Park, Korea, pp.1197–1201.

Balfe, S. and Paterson, K.G. (2008) 'e-EMV: emulating EMV for internet payments with trusted computing technologies', *3rd ACM Workshop on Scalable Trusted Computing,* pp.81–92.

Brett, A., Kuntze, N. and Schmidt, A.U. (2009) 'Trusted watermarks', *IEEE Intl. Symposium on Broadband Multimedia Systems and Broadcasting (BMSB '09)*, pp.1–7.

Brickell, E., Camenisch, J. and Chen, L. (2004) 'Direct anonymous attestation', *11th ACM Conference on Computer and Communications Security*, pp.132–145.

Brickell, E., Camenisch, J. and Chen, L. (2005) *Trusted Computing*, IEE, London, Ch. The DAA Scheme in Context, pp.143–174.

Camenisch, J. (2004) 'Better privacy for trusted computing platforms', *ESORICS 2004,* pp.73–88.

Camenisch, J. and Lysynskaya, A. (2003) 'A signature scheme with efficient protocols', in Cimato, S., Persiano, G. and Galdi, C. (Eds.): *Security and Communication Networks*, pp.268–289, Springer, Berlin/Heidelberg.

Dhamija, R., Tygar, J.D. and Hearst, M. (2006) 'Why phishing works', *2006 Conference on Human Factors in Computing Systems*, ACM, pp.581–590.

Fichtinger, B., Herrmann, E., Kuntze, N. and Schmidt, A.U. (2007) 'Trusted infrastructures for identities', *5th Intl. Workshop for Technical, Economic and Legal Aspects of Business Models for Virtual Goods*, Koblenz, Germany.

Gajek, S., Sadeghi, A-R. and Winandy, M. (2009) 'TruWallet: trustworthy and migratable wallet-based web authentication', *4th ACM Workshop on Scalable Trusted Computing*, pp.19–28.

Gajek, S., Sadeghi, A-R., Stüble, C. and Winandy, M. (2007) 'Compartmented security for browsers-or how to thwart a phisher with trusted computing', *IEEE Conference on Availability, Reliability and Security 2007*, pp.120–127.

Gallery, E. (2005) *Trusted Computing*, IEE, London, Ch. An Overview of Trusted Computing Technology, pp.29–114.

Jackson, C., Simon, D.R., Tan, T.S. and Barth, A. (2007a) 'An evaluation of extended validation and picture-in-picture phishing attacks', *Usable Security (USEC07).*

Jackson, C., Boneh, D. and Mitchell, J. (2007b) 'Transaction generators: root kits for web', *Proceedings of the 2nd USENIX Workshop on Hot Topics in Security (HOTSEC'07),* Boston, MA, pp.1–4.

Kuntze, N. and Schmidt, A.U. (2007) 'Trusted ticket systems and applications', *IFIP SEC 2007*, Sandton, South Africa, pp.49–60.

Kuntze, N., Mähler, D. and Schmidt, A.U. (2006) 'Employing trusted computing for the forward pricing of pseudonyms in reputation systems', *Axmedis 2006: Proc. 2nd Intl. Conference on Automated Production of Cross Media Content for Multi-channel Distribution*, pp.145–149.

Leicher, A. and Brett, A. (2010) *Ethemba, a Trusted Computing Demonstration and Experimentation Environment*, available at http://ethemba.novalyst.de (accessed on 18 August 2010).

Leicher, A., Kuntze, N. and Schmidt, A.U. (2009) 'Implementation of a trusted ticket system', *IFIP SEC 2009*, Pafos, Cyprus, pp.152–163.

Leung, A., Chen, L. and Mitchell, C.J. (2008) 'On a possible privacy flaw in direct anonymous attestation (DAA)', *TRUST 2008*, Villach, Austria, pp.179–190.

OpenID (2010a) *OpenID Specifications*, available at http://openid.net/developers/specs/ (accessed on 18 August 2010).

OpenID (2010b) *OpenID – Year 2009 in Review*, available at http://openid.net/2009/12/16/openid-2009-year-in-review/ (accessed on 18 August 2010).

OpenID (2010c) *OpenID Security Best Practices*, available at http://wiki.openid.net/OpenID-Security-Best-Practices (accessed on 18 August 2010).

OpenID Foundation (2010) *Open Trust Frameworks for Open Government: Enabling Citizen Involvement through Open Identity Technologies*, available at http://openid.net/docs/OpenTrustFrameworksforGovts.pdf (accessed on 18 August 2010).

OpenID4Java (2010) *OpenID 2.0 Java Libraries*, available at http://code.google.com/p/openid4java/ (access on 31 March 2011).

Pashalidis, A. and Mitchell, C.J. (2003) 'Single sign-on using trusted platforms', *ISC 2003*, pp.54–68.

Pashalidis, A. and Mitchell, C.J. (2005) *Trusted Computing*, IEE, London, Ch. Single Sign-on using TCG-conformant Platforms, pp.175–193.

Perrig, A. and Song, D. (1999) 'Hash visualization: a new technique to improve real-world security', *International Workshop on Cryptographic Techniques and E-commerce*, pp.131–138.

Plaquin, D., Cabuk, S., Dalton, C., Kuhlmann, D., Grete, P., Weinhold, C., Böttcher, A., Murray, D., Hong, T. and Winandy, M. (2009) *TPM Virtualisation Architecture*, Project Deliverable IST-027635/D04.7 FINAL 1.0 Update.

Rosenquist, R.M. (2009) 'Method and apparatus for secure mode indication', *United States Patent Application Publication*, US 2009/0144838A1.

Rudolph, C. (2007) 'Covert identity information in direct anonymous attestation', *IFIP SEC 2007*, Sandton, South Africa, pp.443–448.

Sadeghi, A-R. and Stüble, C. (2004) 'Property-based attestation for computing platforms: caring about properties, not mechanisms', *ACM Workshop on New Security Paradigms*, New York, USA, pp.67–77.

Sailer, R., Zhang, X., Jaeger, T. and Van Doorn, L. (2004) 'Design and implementation of a TCG-based integrity measurement architecture', *13th USENIX Security Symposium*, pp.223–238.

Schechter, S.E., Dhamija, R., Ozment, A. and Fischer, I. (2007) 'The emperor's new security indicators: an evaluation of website authentication and the effect of role playing on usability studies', *2007 IEEE Symposium on Security and Privacy*, Oakland, CA, pp.51–65.

Schmidt, A.U., Leicher, A., Cha, I. and Shah, Y. (2010) 'Trusted platform validation and management', *International Journal of Dependable and Trustworthy Information Systems (IJDTIS)*, Vol. 1, No. 2, pp.1–31.

Schneier, B. and Kelsey, J. (1999) 'Secure audit logs to support computer forensics', *ACM Trans. Inf. Syst. Secur.*, Vol. 2, No. 2, pp.159–176.

Smyth, B., Ryan, M. and Chen, L. (2007) 'Direct anonymous attestation (DAA): ensuring privacy with corrupt administrators', *Security and Privacy in Ad-hoc and Sensor Networks*, Springer, pp.218–231.

Strasser, M. (2004) *A Software-based TPM Emulator for Linux*, Semester thesis, Department of Computer Science, Swiss Federal Institute of Technology Zurich.

Third Generation Partnership Project (3GPP) (2010) *Identity Management and 3GPP Security Interworking; Identity Management and Generic Authentication Architecture (GAA) Interworking*, 3GPP TR 33.924, Ver. 9.3.0.

Trusted Computing Group (TCG) (2005a) *TCG PC Client Specific Implementation Specification for Conventional BIOS*, Ver. 1.20, Rev. 1.00.

Trusted Computing Group (TCG) (2005b) *TCG Infrastructure Working Group Reference Architecture for Interoperability (Part I)*, Ver. 1.0, Rev. 1.

Trusted Computing Group (TCG) (2007) *TPM Main*, Specification Ver. 1.2, Level 2, Rev. 103.

Trusted Computing Group (TCG) (2008) *Mobile Reference Architecture*, Ver. 1.0, Rev. 5.

trustedjava (2010) *Trusted Computing for the Java Platform – jTpm Tools and jTSS*, available at http://trustedjava.sourceforge.net (accessed on 18 August 2010).

Tsyrklevich, E. and Tsyrklevich, V. (2007) 'Single sign-on for the internet: a security story', *BlackHat Conference*, Las Vegas, USA.