# Preventing Identity Theft with Electronic Identity Cards and the Trusted Platform Module

Andreas Klenk
Network Architectures and Services
Technische Universität München
klenk@net.in.tum.de

Holger Kinkelin
Network Architectures and Services
Technische Universität München
kinkelin@net.in.tum.de

Christoph Eunicke *
Steria Mummert Consulting
Healthcare / Public Services
Hamburg, Germany
christoph.eunicke@steria-mummert.de

Georg Carle
Network Architectures and Services
Technische Universität München
carle@net.in.tum.de

## ABSTRACT

Together with the rapidly growing number of services in the Internet, authentication becomes an issue of increasing importance. A very common situation is that for each service, users must remember the associated name and password they are registered under. This method is prone to identity theft and its usability leaves much to be desired. The Trusted Platform Module (TPM) is a microcontroller with cryptographic functions that is integrated into many computers. It is capable to protect against software attacks. TPM can generate and store non-migratable keying material for authentication and is an effective safeguard against the acquisition and use of an identity by an adversary. Even though TPM prohibits identity theft, Internet services still have few options to verify the true identity of a user. Electronic identity cards (eID) assert for the identity of their owner. Their large-scale deployment can be expected in the near future. The use of eIDs is impaired, though. They must be present for each authentication, and all devices must be equipped with a compatible card reader. We mitigate the problems of both approaches by using eIDs for establishing trust in user specific TPM authentication credentials. The eID and a compatible reader must be present only at one time for establishing the initial trust. We integrated our identity theft resistant authentication method with the OpenID identity system to allow a large number of services to profit from verified and trustworthy identity assertions.

*This work was performed, when this author was affiliated to the University of Tuebingen.



## Categories and Subject Descriptors

D.4.6 [**Software**]: OPERATING SYSTEMS—*Security and Protection, Authentication*; D.4.5 [**Software**]: OPERATING SYSTEMS—*Communications Management*

## Keywords

Authentication with TPM, Electronic ID, Identity System

## 1. INTRODUCTION

In our modern world, web-based services got omnipresent. The predominating authentication scheme is still knowledge based and relies typically on username and a secret password. This form of authentication leaves much to be desired, because the user must memorize the username/password combination for each account.

Single sign on systems (SSO) address this problem by asserting for authenticated users through an Identity Provider. The user needs now only to log in once, and services obtain the identity and account information from the Identity Provider (IdP). Most Identity Providers lack a reliable method to check the information a user provides when he signs up for an account. For this reason an IdP cannot rule out that an adversary impersonates as a legitimate user. However, many services require a higher confidence in the user identity, for instance, e-commerce and online banking. Furthermore, with SSO systems identity theft becomes an even bigger problem because a single compromised account gives access to a large number of services.

Smart cards allow for a trustworthy execution of authentication functions and other cryptographic operations [1]. With the current government efforts towards electronic identity cards (eID) emerges a new generation of trustworthy smart cards that will have a very broad deployment and can be used for a verified authentication of users towards services or an Identity Provider. The survey in [3] presents current legislative and executive actions in different European countries to distribute trustworthy smart cards with authentication capabilities to the citizens. The usability of smart cards for authentication is inherently impaired: the token needs to

be connected to the device each time the user wants to login to a service. Furthermore, the user is not able to log into services from different devices at the same time.
We want to combine high confidence in authentication with the flexibility of Single Sign On systems. Instead of relying on the constant presence of the eID, we devise a system that uses a cryptographic chip that takes the role of the smart card for authentication. Our method relies on the Trusted Platform Module for identity theft resistant authentication. The Trusted Platform Module (TPM) [13] is a cryptographic chip that is integrated into a large number of computers. The TPM is able to generate and store non-migratable keys in a shielded environment and use them for cryptographic operations. Such "locked-in" keys are not extractable from the TPM by software attacks. Ongoing standardization activities even aim to integrate TPM technology into mobile devices such as cell phones [14].
**Contributions:** This paper introduces a novel system called *TPMident* to establish a high confidence in TPM based authentication with the help of eIDs and to prohibit identity theft. With our approach the relationship is not "one smart card:one identity" anymore, but "many TPM enabled devices:many identities each".
One protocol we designed, utilizes the eID for establishing trust into a key created and stored safely inside the TPM of the device. Once the trust is established into the identity and the device, a second novel protocol authenticates the user towards an Identity Provider using the key stored inside the TPM and the credentials issued by the eID during the trust establishment. This TPM-based authentication method offers a similar level of confidence into the user authentication as with eIDs, but improves the usability and enables more usage scenarios.
We integrated TPMident with OpenID to allow many applications to benefit from trustworthy authentication and to avoid the need to establish trust with each individual service. TPMident can easily be adapted to other identity systems as well. We present a detailed security analysis and performance evaluation of our prototype implementation.
**Outline:** Sec. 2 gives an overview of related work. Requirements follow in Sec. 3. OpenID is introduced in Sec. 4. We first present phishing resistant authentication with the TPM in Section 5 before we introduce a protocol for establishing trust with eIDs in Sec. 6. The security analysis follows in Sec. 7. Sec. 8 gives performance measurements whilst Sec. 9 discusses practicability an future work. The paper concludes in Sec. 10.

## 2. RELATED WORK

There are a number of approaches which study the combination of TPM with smart cards. Gawlas et al. proposed the use of Smart Cards together with TPM in [5]. They argue that the portable smart card complements the immobile TPM chip. They propose to use the smart card to carry the 20 bytes long TPM password and use the smart card to attest the integrity of the platform. In [6] the autor also votes for a complementary usage of TPM and smart card. The TPM assures trustworthiness of the platform, whereas the smart card authenticates the user and carries his credentials. There are approaches that rely solely on the TPM for improving the security of authentication. Stumpf et al. discuss in [12] the applicability of TPM for advanced electronic signature from a juridical perspective. The authors in [2] argue how to augment SSL client-side authentication with TPM functionalities to counter identity theft by storing the private key in the TPM. They do not explain how the certification authority establishes the trust relationship with the TPM system and how it can resolve identities. Latze et al. argue in [7] how the TPM prevents identity theft, because the TPM does not allow the extraction of the sealed keys. The authors explain how to establish the trust relationship with one time passwords by using mobile phones as trusted channels.
Pashalidis and Mitchell presented the use of TPM authentication for single sign on in their research in [9]. They require a full Remote Attestation (refer to 6.1) to guarantee a trustworthy system. The basic assumption of Remote Attestation is that it is possible to know and verify the integrity of all applications and configurations of a system during the Remote Attestation. This assumption is maybe feasible for companies with strict roll-out management of computers and software. However, it is nearly impossible to verify the integrity of a foreign system that can have arbitrary software and normal users probably have privacy concerns to provide a list of all their running applications to a remote party.
All surveyed research has in common that it perceives the single factor authentication with username and password as highly vulnerable. However, none of the presented solutions works well in nowadays heterogeneous environments where mobile phones, computers, laptops and netbooks are used concurrently. The presented solutions that combine TPM and smart card inherit one problem from the smart card: users usually have only one smart card of a certain type and can therefore only authenticate at one device at a time. The TPM-only solutions suffer from the lack of trust in the digital signatures of the TPM. We will present an approach that combines the strengths of smart cards for identity management with TPM authentication, by using the smart card to establish trust in the TPM signatures, and the TPM for two factor authentication.

## 3. REQUIREMENTS

Before we introduce our system, we first present requirements that are particularly important for trustworthy authentication. Authentication can be defined as the act of verifying that claims made by or about a subject are true. We focus on different notions of identity, where the use of an electronic identifier is always restricted to interactions with the same user, but does not necessarily reveal any personal information:

- *Verified Identity* - The information a user presents during registration can be verified reliably. The use of legally binding authentication methods improves the confidence in the registered information.
- *Unverified Identity* - The information a user provides when he registers at an Identity Provider cannot be verified, a user could create a fake identity, for instance.
- *Pseudonym* - The Identity Provider does not reveal any information pertaining to the identity of the user, but guarantees that a specific identifier can be used by a single user only. In this case, an Identity Provider does not need any knowledge about the identity of the user.

For all presented types of identity, it is important to assure that only the registered subject can authenticate with the given identity at a service. For many services it is sufficient to assure that only one particular user can use the identity. Online forums, for example, primarily require authentication to prohibit that an adversary can impersonate as another user.
Devices can also be part of the identity. A user might only be authenticated whilst using a certain device, for example, a particular mobile phone. Sometimes it is sufficient to identify a device without any relationship to a particular user. For realizing these objectives, the Identity Provider should fulfill the following requirements:

- *Unforgeable Identity Assertion* - Only the Identity Provider responsible for the unique identity identifier can make verifiable assertions pertaining the identity.
- *Globally Unique Identifiers* - No party can claim an identifier that has been issued by another Identity Provider.
- *Trustworthiness* - The Identity Provider makes only sincere identity assertions.
- *User Control* - The user must be in control of all information that can be retrieved from the Identity Provider. That includes who can access identity information, what attributes should be released, and if Single Sign On should be used.

The authentication process itself must also assure that certain requirements are met:

- *Device Dependent Authentication* can be performed with the registered device only.
- *User Dependent Authentication* can be performed through authentication of the user only.
- *Device Independence* for User Dependent Authentication - It must be possible for a user to authenticate using different devices. This can mean in the context of device dependent authentication methods that a user registers multiple devices to authenticate for using one identity.
- *Mutual Authentication* - Identity Provider and client authenticate each other and assure the absence of an malicious interceptor of the communication.
- *Key Management* - The service must be able to validate and store keying material for authentication. For validation purposes, the use of a PKI (Public Key Infrastructure) is desirable.
- *Prohibit Identity Theft* - The authentication method should not allow a malicious third party to transfer the authentication credentials to another system and impersonate as a legitimate user.

# 4. OPENID AND AUTHENTICATION

This section gives a brief introduction in authentication with OpenID [8]. The OpenID framework focuses on an URL based user centric protocol for open and lightweight authentication and works without a central authority. The entity that makes identity assertions is called OpenID Provider (see Figure 1). The consumer of the identity assertion is termed Relying Party (RP). The authentication methods themselves are not part of the OpenID standards. Instead, OpenID allows any kind of authentication method.

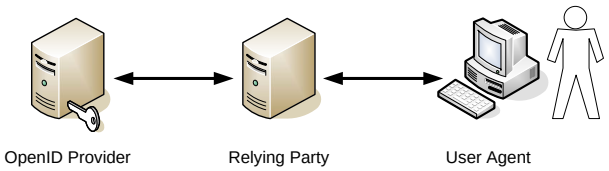


**Figure 1: OpenID Entities**

We based our work on the following OpenID message exchange, as shown in the UML time sequence diagram in Figure 2.

1. The user visits the login page and provides his identifier to the RP.
2. RP sends a browser redirect to the User Agent (UA).
3. The user authenticates at the OpenID Provider with some method.
4. When the OpenID Provider is convinced that the user is authenticated, it generates and signs the final result.
5. The OpenID Provider sends a browser redirect containing result and signature to the UA. UA follows the redirect and delivers result and signature to RP.
6. RP validates the result using the signature.

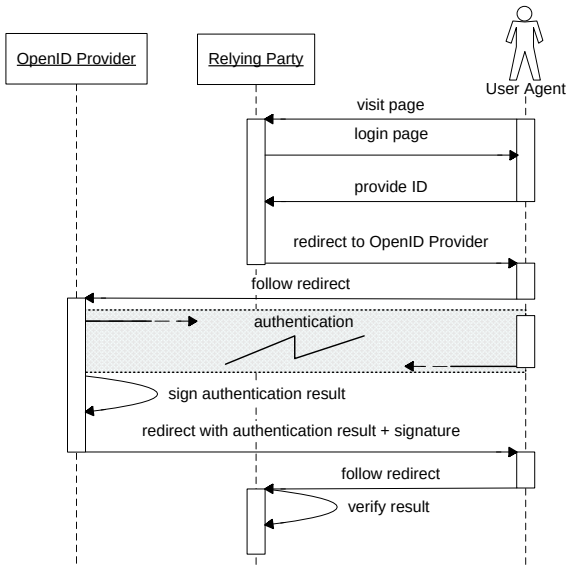


**Figure 2: Example of OpenID message sequence with placeholder for Authentication Method**

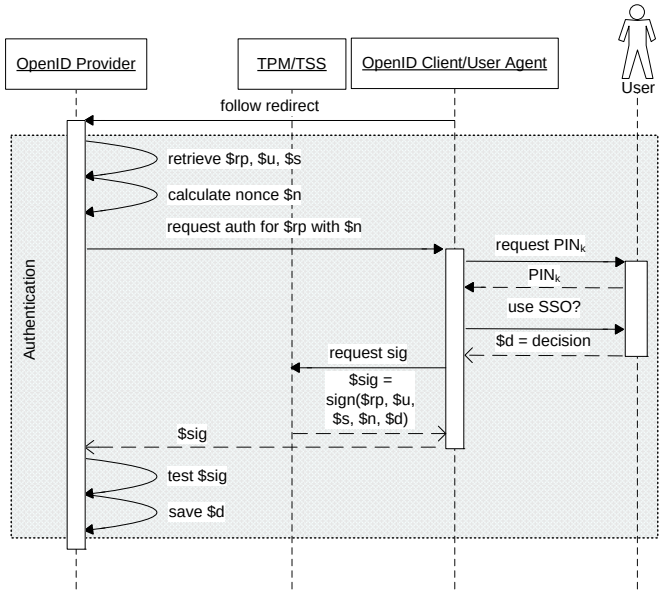


**Figure 3: TPM-based authentication**

# 5. PREVENTING IDENTITY THEFT WITH TPM FOR AUTHENTICATION

Many phishing attacks target identity information and authentication credentials for impersonating the user. We rely on the TPM for mitigating identity theft. The TPM chip allows us to create and lock cryptographic keying material inside the chip, and prevents any attempts in software to compromise critical data of the TPM. Keys can be generated as *non-migratable*, their private key is guaranteed not to leave the TPM then. An optional PIN can control access to the key. Each TPM has one unique, immutable so called *Endorsement Key* (EK) to proof the presence of a genuine TPM. So called *Attestation Identity Keys* (AIK) serve as certified and non-migratable aliases for the EK.
Additionally, we introduce the term Identity Key for a user specific, PIN protected and non-migratable key. The cryptographic operations for the client side authentication happen inside the TPM chip. We thereby assure that authentication cannot work without access to the TPM device. Because the Identity Key cannot be transferred to another platform, most forms of identity theft are not applicable anymore.
As discussed earlier in this paper, TPM authentication alone is no solution for obtaining Verified Identities, because an Identity Provider has no means to relate the platform identity asserted by the TPM with a user identity. The initial trust establishment remains the major hurdle for obtaining Verified Identities with TPM. Refer to Section 6 for our approach to trust establishment with eIDs.

## 5.1 TPM-based authentication for OpenID

As described in Section 4, OpenID can be used with any user authentication method. The TPMident authentication protocol was designed to avoid problems with Network Address Translation. For this reason, all connections originate from the client. Authentication starts when the client follows the redirect to the OpenID Provider (see Figure 2). The client uses HTTPS to establish secure connections to protect against eavesdroppers and verifies the public key certificate of the server. Here is the message sequence for the authentication as drawn in Figure 3:

1. The client establishes a secure TLS connection with the OpenID Provider and verifies the server certificate.
2. The OpenID Provider generates a nonce $\$n$. The URL of the relying party $\$rp$, the OpenID User ID $u, the ID the system is registered under $\$s$ and $\$n$ are sent to the client.
3. The client displays the URL of the Relying Party. The user can choose to discard the request, to authenticate one times for this RP or to activate single sign on to authenticate the user automatically from now on.
4. The client asks the user for the $PIN_k$ to access his Identity Key.
5. The client requests the TPM to sign $\$rp$, $\$u$, $\$s$, $\$n$ and the SSO decision $\$d$ and provides $PIN_k$.
6. The TPM generates $\$sig = sign(\$rp; \$u; \$s; \$n; \$d)_k$ with key $k$. The Signature is sent to the OpenID Provider.
7. The OpenID Provider checks the signature plus nonce and decides if the authentication was successful.

After the identity has been authenticated successfully, the OpenID Provider proceeds with the OpenID Protocol, signs the authentication result and sends it via browser redirect to the Relying Party.
Note that this protocol works even without Verified Identities. It enables Unverified Identities and Pseudonyms, because it guarantees that always the same user/device combination authenticates with this protocol. Device independent authentication with Verified Identities requires a secure mechanism to gain trust that the key $k$ belongs to a specific identity and cannot be compromised.

# 6. ESTABLISHING TRUST

The basic idea of TPMident is to protect against identity theft with TPM authentication, and to relate it with high confidence to a user identity. We will first present Remote Attestation as a tool for verifying system integrity. A discussion of different suitable trustworthy execution environments follows. Finally we introduce a protocol for establishing trust in TPM authentication with eIDs.

## 6.1 Remote Attestation

The process of trust establishment demands a secure execution environment. The basic idea of integrity measurements for security is to determine if a host is running only unmodified and authorized code. Our approach is based on systems in which all code is measured before execution. Measurement values are applied onto a secure TPM hardware register, the so called *Platform Configuration Register* (PCR), and are added to a Measurement List (ML). The Integrity Measurement Architecture (IMA) [10] is an extension to the Linux kernel that performs integrity measures of all code that is loaded into memory for execution and that also can measure arbitrary data upon request. IMA uses the PCRs of the TPM and provides access to its ML. The TPM provides the *extend* operation $PCR_{new} \leftarrow$

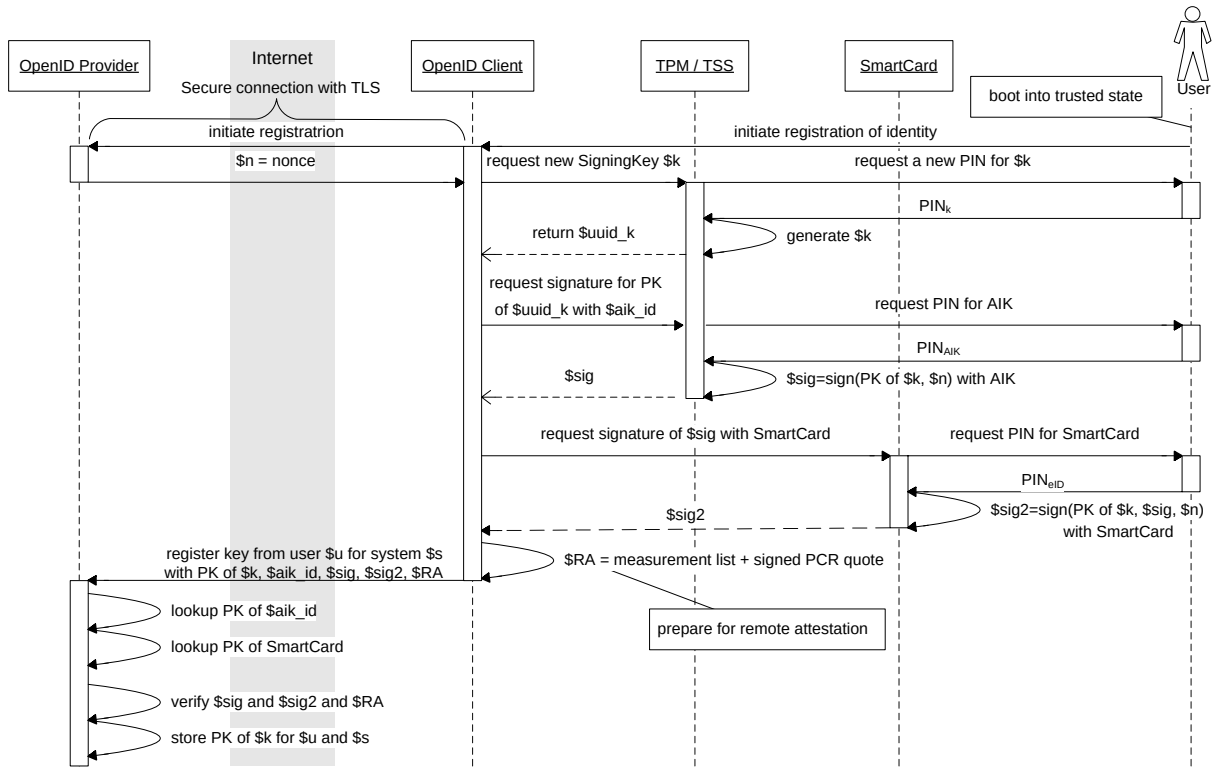


**Figure 4: Protocol for Trust Establishment with eID**

$SHA1(PCR_{old}||M)$ to hash the concatenated values of $PCR_{old}$ and the value from the integrity measurement $M$. The *extend* operation is the only method to change the PCR values. The TPM *quote* operation provides a non manipulable mechanism to sign the PCR values. A remote party that receives the signed PCRs together with the ML can recalculate the PCR value by applying the *extend* operation successively to the entries in the ML. If it arrives at the same value as in the PCR, it knows that the ML is correct. This process is called *Remote Attestation* [10].
The next step is to check that the entries of the ML are certified to run on the system. It ensures that the correct software versions with correct configuration are executed before deeming a system trustworthy. A strict assumption is that the execution of any uncertified code leads the remote party to loose its trust.

## 6.2 Trustworthy Execution Environments

A trustworthy execution environment is crucial during trust establishment. An attacker could exploit an untrusted system to trick the user into signing a fake Identity Key with the eID. The fake Identity Key could reside in another TPM under the control of the adversary, and could be used to impersonate the victim in future. There are different options how to proof the absence of malware with Remote Attestation:

- *Complete Trust* - The Remote Attestation proofed that the system is only executing authorized code and configurations. Any software that is not authorized, will lead to a system being deemed untrustworthy. It is infeasible to ensure complete trust for an arbitrary system, because there are uncountable software variants and configurations, that cannot be all verified. Virtualization and trusted micro kernel systems might mitigate this problem in future.
- *Temporary Complete Trust* - Complete Trust can also be established temporarily, for instance, by using a boot disk. This boot disk contains a minimal trustworthy operating system and only the software needed for establishing trust in TPM authentication. Using Remote Attestation, the device can prove that it has booted in a trustworthy state with the boot disk.

In our opinion, Temporary Complete Trust with Remote Attestation is the preferable option for trust establishment. The required boot disk can be distributed by the TPMident enabled OpenID provider, for instance, as a downloadable vulnerability free boot image. Distribution by the provider ensures that he can certify the complete boot image and verify the complete system by Remote Attestation.

## 6.3 Protocol for Establishing Verified Identities

We stressed already in Section 5 that TPM authentication alone only guarantees that the authentication credentials cannot be stolen. It cannot assure the identity of the user. This section describes therefore a novel protocol for establishing verified identities with eIDs. An Identity Provider uses this protocol to obtain certainty that:

1. the identity key $k$ is stored in a genuine TPM

2. the identity key $k$ was generated by the user

3. the claimed identity belonging to $k$ can be confirmed

TPMident establishes trust by using the eID of the user for signing the public portion of a newly generated Identity Key $k$, which is, as explained in Section 5, user specific, non-migratable and PIN protected. This action establishes trust in $k$, because the user must insert his eID into a card reader and enter his $PIN_{eID}$ number to perform the signature. The eID signature is considered trustworthy, because the government verified the identity of the user when issuing the eID. Access to the Identity Key is restricted by a secret $PIN_k$ number that should only be known by the user. The complete protocol for trust establishment is shown in Figure 4. Here is the textual description of the protocol:

1. *Start up* - The initial action is booting the device into a completely trustworthy state, e.g. by using a boot disk.

2. *Generation of k* - The Client requests a new Identity Key $k$. For this process, the user needs to enter a secret $PIN_k$ that will be requested by the TPM whenever $k$ is used. After the user entered $PIN_k$, the TPM computes the new key and returns the key handle to the Client.

3. *Signing of k* - The newly generated key $k$ needs to be signed with an AIK (or EK) to proof that $k$ resides non-migratably in the TPM. The TPM computes and returns $sign(pub_k, prop_k)_{AIK}$ to the client, which includes the public key and properties of $k$. The properties state that the key is non-migratable and cannot leave the TPM.

4. *Personalization* - $k$ is bound to an identity by signing the public part of $k$ and the previous signature with the eID: $sign(pub_k, sign(pub_k, prop_k)_{AIK})_{eID}$. The user is now requested to enter the personal $PIN_{eID}$ of the eID. The eID computes and returns the signature to the Client.

5. *Registration* - $k$ needs to be registered at the OpenID Server before it can be used by user $u$ on system $s$. The identifyer $s$ might be derived from the EK or be computed randomly. For registration, the client sends $pub_k$, $ID_{AIK}$, $sign(pub_k, prop_k)_{AIK}$ and $sign(pub_k, sign(pub_k, prop_k)_{AIK})_{eID}$ to the OpenID Provider. It appends all required data for a Remote Attestation: the measurement list and the signed quote of the PCRs. The Server verifies the platform integrity and all signatures. It finally stores $pub_k$ for the user in its database.

The final step is to enroll $k$ at the Identity Provider. The key $k$ is now ready for authentication as described in Section 5. A user can register many different devices for the same identity with this method.

# 7. SECURITY ANALYSIS

In the following section, we are going to discuss the security of our solution. Recall that the system has two main objectives: 1.) preventing identity theft 2.) establishing a high confidence in the identity and authentication by a user. Therefore, the attacker model not only includes a malicious third party, but also interior attacks, for instance, by a user who tries to forge identity information. We use attack trees as introduced by Schneier in [11], to systematically analyse threats that our solution protects against and to discover possible vulnerabilities. Children of a node in the attack tree are subgoals to achieve the goal of the parent node. AND nodes denote different steps for achieving the same parent goal, whereas OR nodes indicate alternatives.

```
Tree 1 - Attacks on the TPM itself
AND 1. Exploit TPM Hardware
    OR 1.1. Non functional TPM included in device
       OR 1.1.1 TPM has security vulnerabilities
       1.2. Disclosure of TPM EK private key
       1.3. TPM has hardware backdoor
       OR 1.3.1. controlled by TPM driver
          1.3.2. physical access to TPM
    2. Extract key k from TPM
    OR 2.1. eavesdropping PIN_k
       2.2. extract PIN_k
```

*Assessment:* Attack 1 relies on a broken TPM and can be detected either by running the TPM self-test or by validating the TPM credentials. The whole TPM concept would suffer a serious blow, if a critical security vulnerability of the TPM specification or implementation would be discovered, because it is nearly impossible to replace deployed TPM chips. The disclosed EK private key from attack 1.2. or the Hardware Backdoors of attack 1.3. are nearly impossible to detect. However, these attacks require a TPM manufacturer to risk its reputation and therefore seem to be unlikely.
*Result*: Attacks on the TPM must be considered dangerous but are difficult to perform, since either collaboration of the manufacturer, or technical expertise, technical equipment and physical access is required. This class of attacks appears unlikely for the common Internet user.

```
Tree 2 - Software attacks during trust establishment
OR 1. Malware controls parts of the system
   OR 1.1. Attacker substitutes k with own k' ...
      OR 1.1.1. creation of a key in the user's TPM
         1.1.2. creation of key in adversary's TPM
         1.1.3. creation of an arbitrary RSA key
      1.2 Attacker tricks user in signing key k'
   2. Simulated TPM under control of adversary
```

*Assessment:* If a user wants to establish trust into his infected device with the eID, malware substitutes the Identity Key $k$ of the user with the key $k'$ of the attacker and receives a legitimate signature over $k'$ with the eID. Another option for obtaining a signed $k'$ is social engineering 1.2. to convince the user to sign a "forged" key. It should not be possible to simulate a TPM in software, because the simulated TPM would lack the platform credentials and the private key of the EK.
These attacks are severe, because the user can be tricked into signing a key under the control of the adversary, who could impersonate the user from this time on. Since the most critical part of our process is trust establishment, we already stressed in section 6 the need for a fully trusted system during this process.
*Result:* The use of a "trusted" boot medium combined with remote attestation counters these threats, but still relies on a wary user who withstands social engineering attacks.

```
Tree 3 - Software attacks during authentication
OR 1. Malware controls parts of the system
   AND 1.1. Eavesdropping of the PIN_k
       1.2. Usage of TPM keys ...
       OR 1.2.1. usage of user-generated keys
        AND 1.2.1.1 intercepting PIN_k
        1.2.2. extracting k out of the TPM
       1.3. Man in the middle attack to ...
       OR 1.3.1. substitute the key in the message
          1.3.2. forge the answer of the server
```

*Assessment:* It is important to note, that all above mentioned attacks are reversible after the user gets in full control of the system again. Attacks like eavesdropping of $PIN_k$ 1.1. is generally possible and enables an attacker to *use* $k$ by remotely controlling the system. *Stealing* $k$ still remains impossible. This allows the user to simply change $PIN_k$ after he is in control again. Especially attack 1.2.2. is not possible with software alone, as described above.
The attacker in 1.3. intercepts the communication within the system and could modify messages and substitute keys with his own keys. This attack can be detected by the OpenID Provider by verifying digital signatures and nonces.
*Result*: This class of attacks must be considered dangerous. However, since it is not possible to steal the identity of the user, it is possible for the user to regain control over the identity. In contrast, it is much more difficult for username and password based authentication to regain control.

```
Tree 4 - Exterior Attacks
OR 1. Communication
   OR 1.1. Man in the middle attack
      1.2. Replay Attack
      1.3. Eavesdropping
      1.4. Denial of Service
   2. Malicious OpenID Provider
   3. Theft of the Device
```

*Assessment:* There are a number of attacks that happen outside the system of the user. In attack 1.1 communication with a real OpenID Provider will be intercepted by a man in the middle. The attacks 1.1., 1.2., and 1.3. can be mitigated by using secure end-to-end connections with mutual authentication, and by applying nonces. Denial of Service 1.4. is generally applicable to most systems but provides no possibility for identity theft. However, it will prevent users from authenticating at relaying parties while the DoS lasts. Attack 2. emanates from a malicious OpenID Provider that can make wrong identity assertions. The attacker might impersonate the user and use all accessible services for his benefit. The security of SSO systems mainly depends on the selection of trustworthy Identity Providers. The attack seems to be unlikely, because relying parties will probably only trust well known and reliable Identity Providers.
If the device is stolen in Attack 3. the PIN-protection of the Identity Key prevents abuse. Additionally the Identity Key can be permanently revoked.
*Result:* Identity theft can be evaded by ensuring mutual authentication, secure communication and trustworthy Identity Providers.

# 8. PERFORMANCE EVALUATION

Performance is a crucial factor for the acceptance of real world authentication systems. This section evaluates the performance of the TPMident prototype. All tests were executed 50 times on two machines with low load in the same directly interconnected LAN. The client is based on the JanRain OpenID library [1] and runs on a Dell Optiplex GX620, which is equipped with a TPM V1.2 manufactured by STM, running Ubuntu Linux (6.10) with a modified 2.6.19 kernel. The OpenID Provider PhpMyID [2] was hosted on a IBM R50 Notebook running Windows XP. We present no numbers on the trust establishment phase, as this is a rare event. Trust establishment takes quite some time, due to starting the machine by the trusted boot medium and performing a remote attestation. We used an early prototype of the German eID of the health sector ("Gesundheitskarte") for first experiments on trust establishment.

The authentication protocol in contrast is performance critical. The tests showed, that a whole TPMident protocol exchange takes a maximum of 3 seconds for authenticating an user towards an RP. We also performed tests on the single protocol fragments to identify the most time consuming actions: The processing of an authentication takes on OpenID Provider $t(provider) = 5,28ms$ and Client $t(client) = 181,16ms$. The TPM cryptographic operations contribute with 175 ms to $t(client)$. The redirect protocol for mitigating the NAT problem takes about $t(redirect) = 257ms$. The runtime of the OpenID protocol was $t(OpenID) = 741ms$. For simplicity we assume a user responds within $t(user) = 1sec$ to the SSO dialogue: "Do you want to authenticate at service x?". The TPMident message exchange is strictly sequential, thus the sum of single measurements equal the complete protocol runtime: $t(provider) + t(client) + t(redirect) + t(OpenID) + t(user) = 2.184,44ms$

These numbers indicate that the interaction with the user is the single most time consuming factor. The performance of other user interactive authentication methods are in the same order of magnitude, for instance, entering the password or inserting the smart card and typing the PIN number. Hence we consider the TPMident performance as acceptable for real world usage.

# 9. DISCUSSION

Our approach is more practical than previous approaches (see Section 2). It assumes Temporary Complete Trusted boot only for trust establishment, which works independent of the installed operating system of the user. As the authentication itself works in untrusted environments, it can work with any software of the user, without the need for integrity measurements. In contrast, related work on TPM authentication assumes a Completely Trusted execution environment [7, 9] at all times, which is not feasible for the common user. TPMident works well for scenarios where TPM enabled mobile phones, computers, laptops and netbooks are used concurrently and the constant switching of an eID is unacceptable. An external smart card reader needs only be attached during trust establishment. This allows much more devices to use identity theft resistant authentication, than with the eID alone. We will now assess how well TPMident fulfills the requirements from Section 3 and give directions for future work.

The first block of requirements refers to the different types of authentication assertions. By the coupling of the au-

[1] http://www.janrain.com
[2] http://siege.org/projects/phpMyID/

thentication secret to a legally binding eID card, our system guarantees *Verified Identity*. *Unverified Identity* can be achieved also by binding an digital identity to an Identity Key that couples the identity to the device's TPM. Unverified Identities do not require an eID or remote attestation. *Pseudonymity* should be realized by the Identity Provider. The requirements *Globally Unique Identifiers*, *Unforgeable Identity Assertion*, and *User Control* refer to functionality provided by the Identity Provider and the OpenID protocol. The selection of *trustworthy* OpenID Provider by the Relying Parties is crucial for security. The last requirements refer to the authentication process and the safety of authentication secrets. *Device Dependent Authentication* is realized by the incorporation of Identity Keys into the authentication. As written above, the Identity Key couples a registered identity to a certain device. By binding the authentication credential to the legitimate user's eID card, the requirement of *User Dependant Authentication* is fulfilled. The user may establish trust in an arbitrary number of devices to achieve *Device Independence*. The use of publicly available and verifiable certificates enables *Mutual Authentication* for OpenID Provider, Relying Party and the components on the system of the user. The *Preventing Identity Theft* requirement was already discussed in detail in Section 7. The TPM technology in combination with eID trust establishment protects against identity theft.

There are still some issues left, that should be investigated. OpenID has no notion of trustworthiness. Future work would be an integration of a trust metric with OpenID to express the degree of confidence an Identity Provider has in the authentication. For our prototype, we assume that relying parties acknowledge that our particular OpenID provider asserts for trustworthy and identity theft resistant authentication. Future work is also to improve compliance with the OpenID Provider Authentication Policy Extension (PAPE). TPMident alone does not solve the problem of a malware that infected the systems, intercepts the PIN number and acts as a proxy for an adversary. Virtual Trusted Computing [4] can mitigate this attack. In this case, the authentication service is running in a virtual, completely trustworthy environment that can guarantee the absence of malware. Another interesting question is, if legally binding assertions can be made with TPMident, because the trust in the TPM authentication has been established with an eID that is capable of performing qualified digital signatures.

# 10. CONCLUSION

Identity theft by malware is an urgent threat for Internet services that can be alleviated through authentication with the Trusted Platform Module. Whilst the TPM locks user credentials to the platform and protects against identity theft, establishing trust in TPM authentication still remains an issue. In this paper, we proposed TPMident, a TPM based two factor authentication solution with verified user identities that protects against identity theft. We use electronic identity cards to establish trust in TPM authentication. Our approach is advantageous over authentication with eIDs alone, because it allows for multiple concurrent authentications without the constant presence of a smart card reader or the eID itself. TPMident only requires a trusted environment for the initial trust establishment for one user and one device each. After that, TPMident does not require integrity measurements or other restrictions of the running software during authentication. Our OpenID prototype demonstrated the feasibility for single sign on systems and allows a large number of services to profit from trustworthy authentication. In our analysis we examined the security of the whole process of trust establishment and authentication and showed that the performance of TPM based authentication is in the same order of magnitude than other user interactive authentication methods.

# 11. REFERENCES


[1] M. Abadi, M. Burrows, C. Kaufman, and B. Lampson. Authentication and delegation with smart-cards. In *TACS'91: Selected papers of the conference on Theoretical aspects of computer software*, pages 93–113, Amsterdam, The Netherlands, The Netherlands, 1993. Elsevier Science Publishers B. V.

[2] A. Alsaid and C. J. Mitchell. Preventing phishing attacks using trusted computing technology. In *Proceedings of the 6th International Network Conference*, pages 221–228, july 2006.

[3] S. Arora. National e-ID card schemes: A European overview. *Information Security Technical Report*, 13(2):46–53, 2008.

[4] S. Berger, R. Ceres, K. Goldman, R. Perez, R. Sailer, and L. van Doorn. vTPM: Virtualizing the Trusted Platform Module. Technical report, IBM Research, 2006.

[5] F. Gawlas and U. Stutenbaumer. Combined trusted platform modules and smart card solutions. In *ISSE 2005 Securing Electronic Business Processes: Highlights of the Information Security Solutions Europe 2005 Conference*, pages 92–97. Springer Verlag, 2005.

[6] P. George. User Authentication with Smart Cards in Trusted Computing Architecture. Technical report, Gemplus, 2004.

[7] C. Latze and U. Ultes-Nitsche. Stronger authentication in e-commerce: How to protect even naive user against phishing, pharming, and mitm attacks. In *Proceedings of the IASTED International Conference on Communication Systems, Networks, and Applications*, pages 111–116, october 2007.

[8] OpenID Foundation. *OpenID Authentication 2.0*, 2007. Final Specification.

[9] A. Pashalidis and C. J. Mitchell. Single Sign-On Using Trusted Platforms. *Information Security, 6th International Conference, ISC*, pages 1–3, 2003.

[10] R. Sailer, X. Zhang, T. Jaeger, and L. van Doorn. Design and Implementation of a TCG-based Integrity Measurement Architecture. *13th Usenix Security Symposium, San Diego, California*, August 2004.

[11] B. Schneier. Attack Trees. *Dr. Dobb's Journal*, 24(12):21–29, 1999.

[12] F. Stumpf, M. Sacher, A. Roßnagel, and C. Eckert. Erzeugung elektronischer signaturen mittels trusted platform module. *Datenschutz und Datensicherheit-DuD*, 31(5):357–361, 2007.

[13] Trusted Computing Group, Incorporated. *TPM Specification Part 1 – 3*, 2006. Specification Version 1.2, Rev 103.

[14] Trusted Computing Group, Incorporated. *MTM Specification*, 2008. Specification Version 1.0, Rev 6.