The Integration of Trusted Platform Modules into a Tactical Identity Management System

Anders Fongen and Federico Mancini
Norwegian Defence Research Establishment (FFI)
Emails: {anders.fongen,federico.mancini}@ffi.no

***Abstract*—The use of integrity protection mechanisms from a tactical Identity Management (IdM) system is the focus of this paper. While traditional identity management systems supports authentication, and some also access control, there is still a need for attestation of platform integrity. The proposed solution employs the Trusted Platform Module (TPM) hardware unit to secure the integrity of the software configuration, and to provide cryptographic proof to the IdM system for subsequent attestation of the system's integrity. The communicating parties may elevate their mutual trust on the basis of this attestation.**

I. Introduction

Identity Management (IdM) should support authentication, access control and integrity protection. Through attestation issued by a Trusted Third Party (TTP) the subjects can enter into transactions without the need for direct trust relations.

The Public Key Infrastructure (based on the PKIX recommendations [4]) issues attestation of public keys and supports different authentication operations, but its design has been criticized for its weak scalability, poor interoperability, inadequate data structures and inflexible trust model: The revocation model generates much traffic and hinders cross domain operation, the X.509 certificate has a myriad of extensions and validation models, public keys and subject attributes cannot have different lifetimes, and cross-domain operation is inflexible and overly complicated at the same time.

The Single Sign On (SSO) approach, which may be offered through lightweight models like *SAML SSO Profile*[1], also employs a TTP based attestation model. However, few of these lightweight models offer mutual authentication, only the client is usually authenticated. Mutual authentication should be a requirement as *phishing attacks* are so common on the World Wide Web.

In order to overcome these limitations and to study identity management, authentication and access control in tactical network environments, the *Gismo IdM* was developed. It offers mutual authentication, cross domain operation, access control support, scalable operation, short lived credentials and efficient protocols [2].

This paper will investigate another aspect of trust in identity management systems, namely the integrity of the software configuration in the communicating parties.

When using public key cryptography and certificates, it is essential that the private key residing on the user's host is not compromised or misused. For most signature and decryption operations with key pairs, the private key will be loaded into user memory and used for cryptographic operations. There are two risks related to these operations:

1) The private key may leak outside the controlling process and be misused by unauthorized parties to generate signatures on the owner's behalf.
2) Hostile program code (aka malware) may intercept the cryptographic process and, e.g., change the content of the data which is to be signed.

Risk no. 1 may be mitigated through the use of smart cards which encapsulate the private key and offer an interface to the cryptographic operations. Thus, the private key never leaves the smart card and is consequently less exposed to compromise.

Risk no. 2 is sometimes mitigated through strong separation of activities, as found in *Multi Level Security* (MLS) systems, sometimes through anti-malware software (sometimes called *virus protection*). It can also be mitigated through a *sealing* process, where an approved software configuration is verified during system operation, e.g. the bootstrap.

In the presented approach the *Trusted Platform Module* (TPM) is employed, which is a hardware crypto module widely deployed in most laptop and desktop computers, and which is able to mitigate both risks. It offers a hardware-protected storage for private keys and a shielded environment for cryptographic operations similar to a smartcard. Unlike a smartcard, it is integrated with the platform and is therefore able to offer a mechanism through which the software configuration can be verified during bootstrap. It also offers the connection of the two services so that keys can only be used on systems which have passed their bootstrap verification process.

The contribution of this paper is the extension of tactical IdM systems to include support of TPM-based platform integrity verification in a mobile tactical environment and to issue attestation of a successful verification.

The attests may be used during subsequent authentication and access control by a relying party which benefits from the improved security state. This allows the distinction between trusted users and trusted platforms and may bind the identity of a user to that of the trusted platform at authentication time. In this case, temporary TPM generated keys are used to protect the actual communication and user personal keys are not exposed to any other than the TTP, unlike what happens in [2]. The details of the protocol will be explained during the course of the paper.

The remainder of the paper is organized as follows: In Section II a description of the services offered by the TPM chip will be given. Section III describes the Gismo IdM. Section IV discusses the detailed mechanisms during attestation of TPM keys. Section V presents a small number of related questions regarding our research efforts. Finally, Section VI presents research related to our project, and Section VII concludes the manuscript and outlines further research on this matter.

II. Trusted Platform Module

It is assumed that the reader has some knowledge of the TPM design and is familiar with basic Trusted Computing concepts. However, we will briefly review those relevant to our work in order to define some terminology and notation that will be used in the rest of the article.

The TPM contains several asymmetric key types which are used for different purposes. The *Endorsement key* is created during manufacturing and is never used to sign data, but only to prove to a third party that a genuine TPM is involved in a transaction. This is achieved by using the corresponding key certificate, called *Endorsement certificate* which is (indirectly) signed by well known Certificate Authority like Verisign Corp. The endorsement key is denoted EK and the corresponding certificate $Cert_{EK}$.

Out of privacy concerns, rather than using the EK in actual transactions, the TPM will create another type of key to prove its genuineness without compromising its identity. These are called *Attestation Identity keys*, denoted AIK. These keys must pass a certification process before they can be used, in order to verify that they were actually generated by a genuine TPM. The external certificate authority involved in this process is called *Privacy CA* (PCA), which issues the AIK certificate, $Cert_{AIK}$, to be used later when an AIK signature must be verified. A description of the AIK certification process can be found in [10].

Legacy keys, termed LK, can be created at any time and are the only keys that can be used for both signing and encryption. Legacy keys can be certified by the AIK to prove that they are generated and protected by a TPM.

The *attestation* process consists in signing with an AIK the values stored in the TPM Platform Configuration Registries (PCRs), also called a *Quote* operation, and present them to a third party to verify the platform status. When the PCRs contain the measurements of the software involved in the boot process, we talk about *trusted boot*. A key can also be bound to specific PCR values, so that it can only be used when the platform is in the defined configuration. This is different from *sealing*, which binds encrypted data, not a key, to a specific platform configuration.

There are more types of keys in the TPM, but they are not relevant to the work presented in this paper.

In the following discussion we denote $(Message)_{Key}$ the messages signed with the private part of a Key, and those encrypted with the corresponding public part $Key(Message)$.

III. Gismo IdM

The presence of an identity management system is essential to TPM-enhanced security. The IdM will serve as a trusted third party (TTP) and issue attests which are protected with its digital signature.

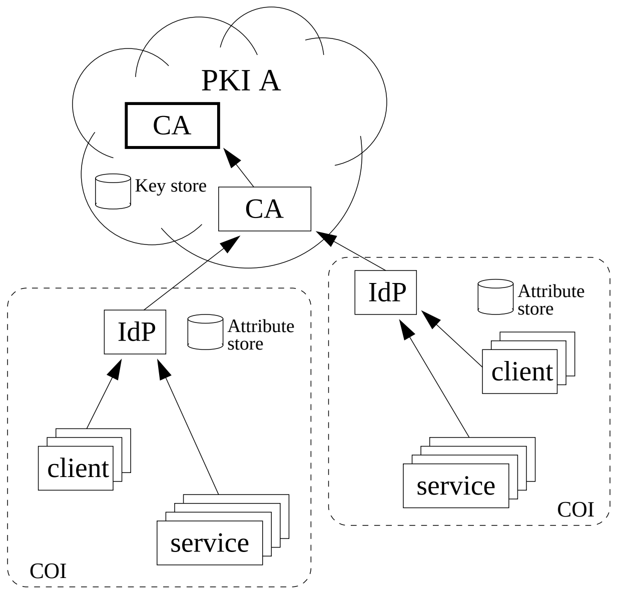

Fig. 1. The functional components of Gismo IdM. Observe that the IdP serves one single COI. Key management is handled by the PKI whereas the attribute management is done by the IdPs on the COI level

For the efforts on establishing a proof-of-concept prototype for attested use of a TPM, an existing IdM called "Gismo IdM" has been employed. Gismo IdM was developed to study the necessary properties for an IdM used in a multi-domain wireless mobile network used by a coalition tactical force.[2]

In Gismo IdM, existing PKIs are kept for reasons of investment protection, but encapsulated by a number of Identity Providers (IdP), each serving a "community of interest" (COI). The members of a COI share the IdP's public key as their trust anchor. The IdP issues *Identity Statements* (IS) to attest the public key and attributes of a subject. Information about the approved software configuration of the subject's computer may be stored among the attributes. The IS is given a short lifetime and sealed with the signature of the IdP. Due to the short lifetime, no revocation arrangement is necessary.

The architectural overview of Gismo IdM is shown in Figure 1. Observe that the COI members are never exposed to PKIX protocols or data objects (X.509 certificates or revocation lists). The key properties are explained in the following paragraphs:

a) Authentication support: The IdP issues ISes which bind the public key of a subject to its identity, analogous to X.509 Certificates. Identity statements are issued to local subjects registered in the IdP, as well as to subjects who can display an IS issued by a different IdP to which this IdP has a *trust relationship*.

The subjects (either client or server) authenticates themselves during the service invocation by the use of their identity statements and their private keys. Different protocols have been designed with the purpose of generating as little network traffic and as few protocol round trips as possible.

b) Integrated access control: Included in the identity statement is a set of attributes which describes properties of the subject in the form of name-value pairs. The attributes

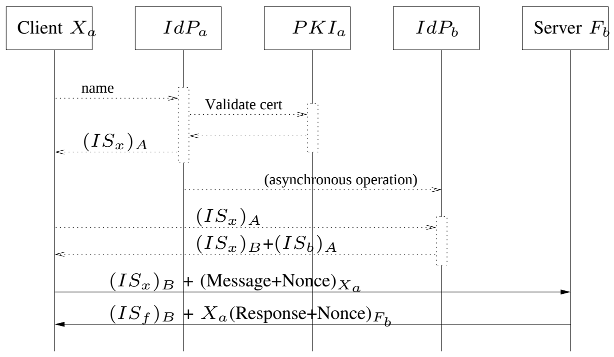

Fig. 2. Gismo IdM protocols for IS issue and service invocation

can describe *roles* of the subject and enter into access control decisions based on the Role Based Access Control (RBAC) or Attribute Based Access Control (ABAC) model. They can also describe other properties of the subject, e.g., preferred language, proficiency level etc. which are utilised during service execution.

The attributes are sealed inside the identity statement with the signature of the IdP, so they cannot be changed once issued.

c) Cross-COI operations: Clients can invoke services in a different COI, provided that there exists a trust relationships between the two COIs. A client obtains an identity statement from its IdP, then passes on that IS to the IdP of a foreign COI. The foreign IdP can issue a *guest IS* containing the same information, but signed by the foreign IdP. Since the guest IS's signature will be trusted by servers in the foreign COI, it can be used to authenticate to these servers. Server authentication requires a *cross domain IS* issued from one IdP to the other, so a signature chain back to the client's trust anchor can be constructed. An example is given in Figure 2 which is to be interpreted as follows:

An Identity Statement IS_x is issued and signed by the IdP_a with its key A to some user X_a after his certificate was verified with the PKI_a. IdP_b gets its IS signed by IdP_a as a proof of mutual trust. X_a wants to use a service in the COI of IdP_b and asks for the guest IS $(IS_x)_B$, which is trusted thanks to $(IS_b)_A$ generated previously. X_a then authenticates itself with the target service on server F_b. The request contains the client guest IS and the message is signed with X_a private key to protect its integrity. Replay is not considered as a threat for this particular service. The response message contains the server's IS (IS_f) and is encrypted with X_a public key as a part of the authentication scheme (not for confidentiality), and signed by F_b to protect integrity. $(IS_b)_A$, which was issued with the guest IS is necessary to validate the server's IS. The Nonce is uses to protect against response replay.

The IdP is not invoked for each service invocation. The IS is cached with the subject for its lifetime and used in all subsequent request or response messages. The IdP does not need to be reachable all the time, only when a new IS is needed. During a tactical operation, this has great advantages. The IdP can be running on a computer in a vehicle, and the soldiers can get fresh ISes prior to dismounting and use these for several hours. For the same reason, the IdP is not an attractive target for a DOS attack of short duration.

IV. TPM-ENHANCED IDENTITY MANAGEMENT

An identity statement can indicate if the computer in use is TPM protected, which means: (1) That it has booted with an approved software configurations, and (2) that the operations which employs the private key take place inside the TPM. Hence, using a TPM can significantly increase the overall security of the system and strengthen the trust between entities. This implies also that the IS, which is issued on behalf of a subject (e.g., a person) now contains information about the computer through which the subject is operating. Consequently, the identity statement must be bound both to the computer and the subject.

The IS issuing process for a TPM client in the IdM consists of two different operations, as shown in detail in Figure 3:

1) *AIK certification*, which is done once by an administrator before the computer is deployed in the field.
2) *LK certification*, whereby a temporary Legacy Key is attested by the IdP through an IS which binds the LK to the identity of the user who currently operates the computer.

Observe that it is the LK certification that results in an IS that subsequently goes into the authentication process. Since a computer is used by several users, the lifetime of the LK will never be longer than the computing session owned by a user. The new user will need a new LK and a new IS in order to operate.

A. AIK certification

In Gismo IdM, public key certificates are issued and kept within the PKI, not the IdM. The certificates are issued as a part the subject's enrollment process, and as a TPM-equipped computer is entered into the equipment inventory its AIK should be validated and certified by the PKI CA.

A newly generated AIK is sent together with $Cert_{EK}$ and a $Quote$ of the 12 first PCRs to the PKI CA, encrypted with the PKI CA public key. The AIK certification process now takes the following steps:

1) $Cert_{EK}$ is validated by following the certificate chain to the root CA (Verisign) according to the outline set by RFC3280.[4] (This is currently possible only with Infineon TPM chips which are shipped with Endorsment Certificates signed by Versign).
2) An AIK certificate is generated, in which the AIK is bound to a subject name which is constructed from the $Cert_{EK}$ serial number and the PCR values. A "handle" (random string) is hashed and the resulting hashed value is associated with the certificate. Other fields and extensions in the certificate are given standard values. The AIK certificate, $Cert_{AIK}$, is signed by the CA and stored in the certificate repository.
3) What is returned to the client is not the AIK certificate, but the "handle" with which it is associated. This string is encrypted with EK, so that only the TPM later can present this handle. Observe that the handle cannot be constructed through inspection of $Cert_{AIK}$ due to the hash operation.

The AIK certification process is logically identical to the standard request to a PCA.[10] The main difference is that $Cert_{AIK}$ is not returned to the client and that the $Quote$ operation usually done in a separate attestation process, is now part of the AIK certification request. A more detailed discussion is given in Section V.

B. Legacy key (LK) certification

Prior to invocation of services, the TPM-equipped computer must obtain an IS which certifies an LK generated by the TPM which will be used for authentication. The LK represents a computer *user*, not the computer itself. In a multi-user system, several LKs may exist and must be reserved to one process/user and protected from use by other processes.

The certification of an LK is carried out by the IdP, and results in an IS with the public part of LK and the identity information from the computer user. The structure of the LK certification request must be able to ensure: (1) That the LK is made inside a genuine TPM, (2) that the platform integrity is guaranteed to some extent, (3) that the user requesting the IS is authentic and authorized. The first two requirements are satisfied by using the `TPM_CERTIFY_INFO` structure generated by the TPM when certifying a new key. This structure contains, among other things, a digest of the LK and the PCR values under which this specific key is allowed to operate, all signed with the AIK. The third is satisfied by binding user and LK by signing the request with the user's private key. The syntax of the request is as follows:

$$(SN, LK, AikHandle, (LKdigest + LK\{PCR\})_{AIK},)_s \qquad (1)$$

which reads as follows: The *digest* of the LK together with the PCR values specified at its creation are signed by the AIK and concatenated with the "handle" of the AIK certificate, so that the IdP knows which $Cert_{AIK}$ to fetch from the PKI repository to verify the signature. The user's subject name (SN) is bound to the LK through the user's signature s and the user's certificate in the PKI.

This request structure is validated and the necessary trust is established, after which the IdP issues an identity statement with the following content:

$$(SN, LK, Attr, ValPer, IdpName)_{IdP} \qquad (2)$$

The terms *Attr* means the subject's attribute set, the term *ValPer* refers to the validity period. The attribute set will contain additional information that the IS relates to a TPM-assisted operation and whether the PCR values are still the same as those quoted in the pre-deployment phase. Except for these extra attributes, the resulting identity statement are identical for TPM-based and normal clients and are compatible with all protocols presented in [2] and [3].

V. Discussion of the proposed solution

There are a number of other aspects on the integration of TPM and IdM that will be discussed briefly in this section:

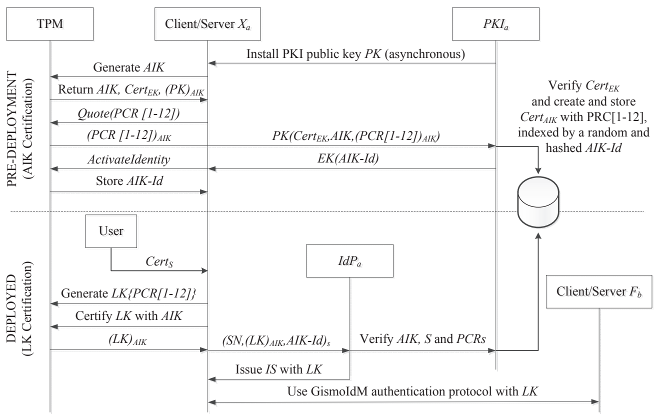
TPM
Client/Server X_a
PKI_a
PRE-DEPLOYMENT (AIK Certification)
Install PKI public key PK (asynchronous)
Generate AIK
Return AIK, $Cert_{EK}$, $(PK)_{AIK}$
$Quote(PCR [1\text{-}12])$
$(PCR [1\text{-}12])_{AIK}$
$PK(Cert_{EK}, AIK, (PCR[1\text{-}12])_{AIK})$
Verify $Cert_{EK}$ and create and store $Cert_{AIK}$ with PRC[1-12], indexed by a random and hashed $AIK\text{-}Id$
$ActivateIdentity$
$EK(AIK\text{-}Id)$
Store $AIK\text{-}Id$
DEPLOYED (LK Certification)
User
$Cert_S$
IdP_a
Client/Server F_b
Generate $LK\{PCR[1\text{-}12]\}$
Certify LK with AIK
$(LK)_{AIK}$
$(SN,(LK)_{AIK},AIK\text{-}Id)_s$
Verify AIK, S and $PCRs$
Issue IS with LK
Use GismoIdM authentication protocol with LK

Fig. 3. This Figure shows in detail the protocol for AIK and LK certification explained in Sections IV-A and IV-B. In a pre-deployment phase an administrator generates an AIK, quotes the PCRs from 1 to 12 ($(PCR[1-12])_{AIK}$), and sends them together with the $Cert_{EK}$ to the PKI encrypted with its public key (PK). The $Cert_{EK}$ is verified and an AIK certificate ($Cert_{AIK}$) containing the PCR values is created and stored. Its identifier ($AIK - Id$) is randomly generated and sent back encrypted with the EK to X_a, where it is decrypted by calling the $ActivateIdentity$ function on the TPM, and stored. The same $AIK - Id$ is also hashed and stored in $Cert_{AIK}$, so that obtaining the certificate would not reveal it. Once deployed, some user logs in with its certificate ($Cert_S$) and a Legacy Key (LK) is generated to be used only with the current PCR[1-12] ($LK\{PCR[1-12]\}$). The key is then signed with the AIK and sent to the IdP with an IS request, together with the user Subject Name (SN), and the $AIK - Id$, all signed with the user key S. Once the signatures and the PCR values are matched against those stored on the PKI, the IS is issued and authentication continues as in Figure 2.

a) TPM, TSS and Java: Gismo IdM has been implemented in Java, and it was therefore a natural choice to implement also the TPM related code in the same language for best compatibility. Fortunately, the JSR-321 [11] was recently approved for a final release and provided us with a TSS (TPM Software Stack) completely implemented in Java. However, these libraries lack the support for some important TPM features like AIK certification and legacy key creation and use. We have therefore implemented those parts ourselves by reusing parts of the jTSS libraries [5], upon which the JSR-321 reference implementation is based. In particular, the AIK certification part was the most tricky as there is no functioning PCA around and the whole process is often only simulated locally in available APIs.

The existing TPM protocols are mainly focused on privacy, but if privacy is not a concern compared to other security properties in the communication between a TPM client and another party, there are several obstacles to overcome: First of all, being able to sign with the EK would greatly simplify the use of a TPM as it would eliminate the need of a PCA and AIKs, as any client could directly verify the EK signature by using well known CAs like Verisign. However this is made impossible by design. Still, AIKs might be useful in a multi-user environment. Secondly, the `Tspi_TPM_CollateIdentityRequest` method specified in [13] requires the encryption of the TPM output to a `TPM_MakeIdentity` command (to create an AIK) with the PCA public key. This implies that it is impossible to modify the request for use with a custom protocol like ours without modifying the TSS itself, which consequently was done. By translating TPM generated data structures into simple Java objects there was no need to handle TPM specific data types like `TPM_KEY` or `TPM_IDENTITY_CONTENTS` on machines without a TPM, like the IdP or in the PKI. Everything can simply be seen as encrypted and signed Java objects, and the AIK and other TPM keys also are treated as normal RSA key objects. This simplified enormously the integration of TPM support in Gismo IdM.

b) AIK and LK certification: A detailed analysis of the certification process is described in Section IV.

Unlike standard AIK certification processes, the $Cert_{AIK}$ is not returned after a successful certification of the AIK on the PKI. If the client had the certificate, then it would always have to send it along with an AIK signature to allow for verification. Besides using more bandwidth, frequent verification of unknown certificates can be quite problematic in a tactical environment where connectivity to the trust anchor (IdP) might not always be available. Also, AIK certificates themselves are not the easiest certificates to deal with, as noted in [6]. In the presented solution the AIK signatures are used only when certifying an LK, hence only the IdP needs access to the $Cert_{AIK}$.

The $Cert_{AIK}$ itself does not need to be verified again, because it can be assumed that if a client sent a certain AIK handle, then a real TPM decrypted the corresponding AIK certification response, and it is known that the corresponding $Cert_{EK}$ was valid since the handle was generated.

In case an attacker without an approved TPM obtained a valid $Cert_{EK}$, he could fake an AIK request and fool the PKI into generating an AIK certificate with an AIK which was under the control of the attacker and not a valid TPM. However, as long as the attacker is unable to decrypt or guess the $AikHandle$, he cannot request a LK certification and obtain a valid IS. Assuming the attacker does not have the EK private key, we have to make sure he cannot obtain the $AikHandle$ by other means. This is why the $AikHandle$ is a completely random value, so that it is nearly impossible to guess, and why we hash it before putting it in the $Cert_{AIK}$. This prevents the attacker from discovering this value even if the newly generated $Cert_{AIK}$ were to leak from the PKI CA. Letting the real TPM where the $Cert_{EK}$ comes from decrypt the $AikHandle$, and then steal the secret value, should also be infeasible. In fact, a genuine TPM will not release the decryption key unless the AIK that is being certified was actually generated and currently loaded in the TPM.

Finally, when certifying a LK the TCG recommendeds the use certificate signing requests (CSR) using a special X509v3 extension called SKAE [12]. This turned out to be almost impossible as there are no standard libraries to create and process SKAE extensions. The choice was made to simply translate the `TPM_CERTIFY_INFO` data structure containing the hash of the LK being certified, the PCR values given as parameter at creation time and the AIK signature, into a simple `SignedObject` in Java.

c) Platform integrity: Platform integrity is usually verified through remote attestation, where the PCR values written during a Trusted Boot process are signed with an AIK and sent with a log file and the $Cert_{AIK}$ to a third party, which in turn verifies the validity of the certificate, the integrity of the PCR values and the system configuration against a database of approved configurations. This is impractical in a tactical mobile environment, as these databases would pose the same challenges as PKI revocation lists. The proposal is to leverage on the fact that machines are in a trusted state when the administrator generates the AIK in the pre-deployment phase. Combined with the fact that tactical missions have a relatively short duration, the assumption can be made that this trusted configuration will remain stable throughout the mission. Thus, the LK can be bound to the current PCR values, assuming that they are still the same as in the AIK certification phase. The verification will take place on the PKI rather than locally or on other clients, thanks to the $Cert_{AIK}$ stored there. In this way there is a guarantee that the LK can be used only if the platform has been booted with a configuration trusted by the administrator. In fact, even if a compromised platform creates an LK with the correct PCR values to obtain an IS, it will not be able to use it since those values are not the current ones. The LK is defined as ephemeral, so that trying to reboot with a valid configuration after getting the IS, would delete the key. The advantage is also that a $Quote$ operation is not necessary, since the interesting PCR values are those used to create the LK, and they are already signed in the `TPM_CERTIFY_INFO` structure. If the PCR values of the LK are different from the stored one, the IdP can either refuse to issue an IS, or issue it with a warning.

At the moment the PCR 1 to 12 are used as they are automatically calculated by any machine with a BIOS supporting a TPM and running Windows 7 or higher, as they are also used to support the BitLocker security features.

d) Compatibility with non-TPM equipped computers: The structure of an IS is the same, regardless if it has been issued through an LK attestation process or an ordinary IS issuing process. As a consequence, all TPM-specific program code is used in the certification phases, while all code related to authentication and access control is not TPM-specific. TPM equipped nodes and non-TPM nodes can communicate without reservations. Since all TPM generated data structures and requests are encapsulated in ordinary Java objects, the communication with the IdP and PKI is seamless with the same interface for cryptographic operations, but with different providers. This puts the burden of parsing, extracting and filtering of the TPM specific information on the TPM-equipped client, which already has specific libraries for this purpose. Please observe that in the spirit of the original Gismo IdM no certificates are ever exchanged. Hence, adding TPM support does not require a new certificate infrastructure or extra PCA, and traffic and protocols between clients are unaffected.

Cross COI operations also still work as described in Section III. Since the IS resulting from an LK attestation process is similar to any other IS, it may also be used for a request of a *guest IS* issue. The guest issue will contain the same information regarding the TPM protection of the client, which may enter into the access control decisions in the foreign COI. No new trust relationships are introduced.

e) Discarded computers: If a computer is stolen or discarded it should be excluded from use. By simply deleting the associated AIK certificate from the PKI certificate repository, no more LK attestation can take place from that computer.

VI. Related research

Research concerning the use of tamperproof hardware units to strengthen the trust in different kind of information systems has been going on for quite some time. The focus of this paper is on Identity Management Systems and in particular on how to increase the trust between users, nodes and services in a dynamic and heterogeneous tactical environment.

The idea of using a TPM to improve Identity Management Systems is not completely new. The use of Trusted Computing (TC) concepts with Single Sign On (SSO) systems has been investigated in [8], [9], while in [7] a TPM based protocol for OpenID is presented. The basic idea in [8] is fundamentally different from our work, since the entity making use of a TPM to attest its trustworthiness is the IdP generating the tickets (identity statements in our case) rather than the client using them. The presented approach facilitates cross domain authentication because the recipient of the ticket can verify the status of the IdP that issued it and therefore decide to trust it without a direct trust relation. A similarity with our system is that also in our case the same ticket (IS) can give access to different services as long as the IdP that issued it is trusted. In [7] the focus is shifted to the client. In this case the ticket is generated directly on the client and signed with a TPM certified key. In [7] the IdP verifies the signature on the ticket, the authenticity of the AIK that certified the signing key, and the status of the client platform that was attested through specific PCR values against a database of valid configurations. If the verification is successful, the client is redirected to the desired service through a request signed by the IdP. So the IdP must be contacted every time the user wants to use a new service. In this case the similarities with our system stop with the generation of an AIK and a TPM certified key on the client side that are later verified by the IdP. In a sense both works bind and delegate the identity of the user to that of the TPM.

The main difference between our solution and previous works about trusted IdM systems is that the intended operational environments are not the same. While OpenID and Kerberos are mostly intended to simplify web services authentication, Gismo IdM is designed to provide both authentication and access control in a tactical environment with relatively low resources, intermittent connectivity and a well defined community of interest. As a consequence, the underlying communication protocols are essentially different. Another difference is that one of the motivation for the use of a TPM in [7] was to improve user experience on the web by eliminating the need for passwords in the authentication process. Passwords are in fact substituted by TPM-protected user certificates, but no guarantee is given that the local protection is sufficient and that a user's AIK cannot be misused. In a tactical environment where different soldiers can use the same communication terminal, this is an important aspect of security. In the Gismo IdM protocol proposed in this paper the TPM keys are cryptographically bound to a user identity, so that we can achieve a higher degree of integrity assurance. In fact, both the platform and the user must have been subject to some verification process before the identity statement is issued. This gives higher flexibility for access control, which can depend on both the user attributes and the degree of trust of the machine used for the communication. Finally, while in [7] the TPM involvement is limited to the certification process which involves the IdP, we extend the use of the TPM certified key to the transactions with the target service, hence improving the security of the system as a whole.

Finally, we are not aware of similar works regarding the lighter platform integrity verification we proposed.

VII. Conclusions

The research efforts presented in this paper resulted in a novel solution to strengthen the trust between peers in a tactical environment with the combined use of TTP agents and hardware protection units. The prototype we developed and the preliminary tests showed that a seamless integration of TPM-equipped machines in the existing Gismo IdM is possible and fully supported by the current infrastructure.

The client libraries of the Gismo IdM is ported to the Android platform and this port is fully interoperable with the TPM extensions presented in this paper.

Future research will include the investigation of a way to extend the trust in the system beyond the boot process, possibly by using a Trusted Execution Environment (TEE) for security critical operations, and a safer way to employ the subject's private key in the LK certification process.

References

[1] John Hughes et. al. *Profiles for the OASIS Security Assertion Markup Language (SAML) V2.0.* OASIS Standard, 2005.

[2] Anders Fongen. Architecture patterns for a ubiquitous identity management system. In *ICONS 2011*, Saint Maartens, Jan. 2011. IARIA.

[3] Anders Fongen. Federated identity management for android. In *SECURWARE 2011*, Nice, France, July 2011. IARIA.

[4] R. Housley, W. Polk, W. Ford, and D. Solo. *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*. RFC Editor, United States, 2002.

[5] IAIK TU Graz. Trusted Computing for the Java Platform. http://trustedjava.sourceforge.net/. Online, Accessed Februar 2013.

[6] Carolin Latze and Ulrich Ultes-Nitsche. A proof-of-concept implementation of eap-tls with tpm support. In Hein S. Venter, Mariki M. Eloff, Jan H. P. Eloff, and Les Labuschagne, editors, *ISSA*, pages 1–12. ISSA, Pretoria, South Africa, 2008.

[7] A. Leicher, A.U. Schmidt, Y. Shah, and Inhyok Cha. Trusted Computing enhanced OpenID. In *Internet Technology and Secured Transactions (ICITST), 2010 International Conference for*, pages 1 –8, nov. 2010.

[8] Andreas Leicher, Nicolai Kuntze, and Andreas U. Schmidt. Implementation of a trusted ticket system. In Dimitris Gritzalis and Javier Lopez, editors, *SEC*, volume 297 of *IFIP*, pages 152–163. Springer, 2009.

[9] Andreas Pashalidis and ChrisJ. Mitchell. Single sign-on using trusted platforms. In Colin Boyd and Wenbo Mao, editors, *Information Security*, volume 2851 of *Lecture Notes in Computer Science*, pages 54–68. Springer Berlin Heidelberg, 2003.

[10] Martin Pirker, Ronald Toegl, Daniel M. Hein, and Peter Danner. A privacyca for anonymity and trust. In Liqun Chen, Chris J. Mitchell, and Andrew Martin, editors, *TRUST*, volume 5471 of *Lecture Notes in Computer Science*, pages 101–119. Springer, 2009.

[11] Ronald Toegl, Thomas Winkler, Mohammad Nauman, and Theodore W. Hong. Specification and standardization of a java trusted computing api. *Softw., Pract. Exper.*, 42(8):945–965, 2012.

[12] Trusted Computing Group. TCG Infrastructure Workgroup Subject Key Attestation Evidence Extension. http://www.trustedcomputinggroup.org/. Online, Accessed February 2013.

[13] Trusted Computing Group. TCG Software Stack (TSS) Specification Version 1.2skar. http://www.trustedcomputinggroup.org/. Online, Accessed Februar 2012.