A Device Management Framework for Secure Ubiquitous Service Delivery

Adrian Leung* and Chris J. Mitchell
Information Security Group, Royal Holloway, University of London
Egham, Surrey, TW20 0EX, UK
{A.Leung, C.Mitchell}@rhul.ac.uk

Abstract

In a mobile ubiquitous environment, service interactions between a user device and a service provider should be secure, regardless of the type of device used to access or consume a service. We present a Secure Device Management Framework (SDMF), designed to securely deliver services to user devices, whilst also hiding (some of) the complexity of security management from users. Key to this framework is the Device Management Entity (DME), that manages a user device's security credentials, and interacts with service providers on its behalf. This framework also provides users with assurance that a compromised device cannot consume the delivered service, and, at the same time, prevents users from illegally sharing their credentials with other users. We achieve these objectives using Trusted Computing functionality and certain other security mechanisms.

1. Introduction

In a mobile ubiquitous environment (such as that shown in Figure 1), users typically own disparate devices (e.g. PDAs, laptops, mobile phones, etc.) with varying form factors and capabilities, attached to a range of access networks, and used to consume a variety of content and services offered by service providers. It is extremely important to secure the service interactions between user devices and service providers. However, achieving this presents many interesting challenges. For example, a user may own devices of many different types, and the device used to access a service may depend on the user's physical location or context. The user just wants to be able to access a service in a simple and secure manner using an appropriate device without being burdened with (too much) technical complexity. A service provider may be happy to allow a user to access services if they have a legitimate subscription (i.e. using a properly issued credential) and the user device is not in a compromised state. The latter task is further complicated by the current landscape, where security threats (e.g. viruses, malware, trojans, spams, etc.) abound. In this paper, we aim to address these challenges collectively; to the best of our knowledge, there is no prior published work on this topic.

*This author is supported by a British Chevening/Royal Holloway Scholarship

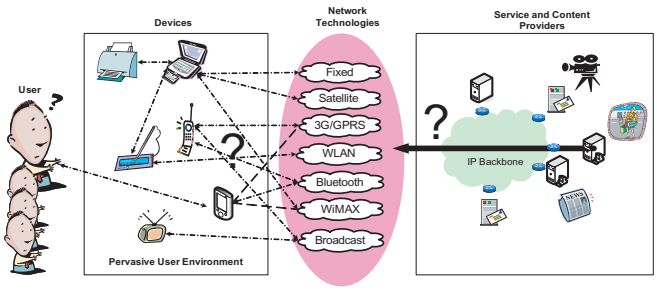

Figure 1. A Mobile Ubiquitous Environment

In this paper we propose SDMF, a Device Management Framework for secure delivery of ubiquitous services to end user devices. Apart from providing secure service interactions amongst the players, the framework is also designed to reduce the complexity of device security management tasks for users. Furthermore, the framework protects the interests of service providers by preventing unauthorised credential sharing amongst user devices. One other novel feature of the framework is that compromised devices are self-revoking, hence removing the need for a cumbersome revocation infrastructure. We achieved these objectives by incorporating Trusted Computing functionality into an entity known as the Device Management Entity (DME), and then integrating it with a number of other security mechanisms (such as the MANA protocol [3] and the Ninja Service Discovery protocol [6]).

The remainder of this paper is organised as follows. In Section 2 we identify the major challenges to secure service delivery. Section 3 gives an overview of the building blocks that we employ to develop the secure device management framework. In Section 4 we present our secure device management framework, and Section 5 analyses its security. Finally, in Section 6, we draw conclusions.

2. Security Issues

We identify the security threats to users and service providers arising in the delivery of services in a mobile ubiquitous environment. We then give a corresponding set of security requirements designed to mitigate these threats.

2.1 Security Threats

The security threats can be summarised as follows:

- **Spoofing:** Malicious entities may masquerade as legitimate service providers by broadcasting fraudulent service messages to users in an attempt to initiate a bogus service interaction. Similarly, malicious entities may masquerade as service users by replaying old service messages (captured earlier) to service providers.

- **Tampering:** Service messages (e.g. service advertisements, registration messages, or reply messages) sent between users and service providers via wireless channels may be overhead by a passive adversary, and modified or deleted by an active adversary. In addition, an adversary may tamper with the service or content information requested by a user whilst the data is in transit. Tampering includes sending (injecting) false service messages or malicious content to users.

- **Information Disclosure:**

    1. **During Communications:** Service messages may be overheard whilst in transit (the risk is particularly high when using a wireless communications channel). This may reveal sensitive information about the user (e.g. the type of service being requested, the service providers a user interacts with, the exact times or dates when services are being accessed, etc.). A user's privacy may thus be compromised. Similarly, an eavesdropper may attempt to intercept the transmitted service/content for his/her own consumption.
    2. **Physical Loss of Device:** Sensitive personal information (such as contact information, emails, etc.) may be stored in user devices. The devices may also hold the security credentials that are used to access subscribed services. Thus, if a user device is lost or stolen, an adversary may be able both to obtain sensitive information about a user and gain unauthorised access to services.

- **Malicious software attack on user devices:** A user device, although still in the physical possession of the user, may be infected with malicious software (e.g. viruses, trojans, etc.). Such software could perform a variety of attacks, including infecting a device, stealing a user's credentials or personal information, or encrypting user data on the device and then demanding payment for the release of the decryption key. A user might be oblivious to such an attack, as his/her device may still appear to be functioning "as normal".

- **Illegal Credential Sharing/Distribution:** Having legitimately obtained the access tokens or security credentials necessary to access a service or content, a user could illegally share the credentials (e.g. with friends or family), possibly for financial gain.

2.2 Security Requirements

Building on the threat analysis in the previous section, we now give a corresponding set of security requirements designed to address these threats.

- **Entity Authentication:** In general, if they use an insecure channel, communicating entities need to authenticate each other to prevent spoofing attacks. First, users should authenticate themselves to their devices, e.g. using a PIN or password, to address the threat of information disclosure arising from physical loss of a device. Secondly, devices and service providers should be mutually authenticated to each other.

- **Integrity Protection:** Messages exchanged between communicating entities (service management messages or the actual service/content) should be integrity protected to address data manipulation attacks. Possible integrity protection mechanisms include Message Authentication Codes (MACs) and digital signatures.

- **Service Confidentiality:** This addresses information disclosure threats. Messages (service management or service messages) sent between entities need to be protected against eavesdroppers and active attackers. Symmetric encryption algorithms using a shared secret key can be used to provide this service.

- **Device Integrity Assurance:** Preventing software attacks on a device is particularly challenging. An alternative to prevention would be to detect whether a device has deviated from specific trustworthy software states. It would be even better if access to service could be prevented if a device has been compromised. This could be achieved using the attestation mechanisms provided by Trusted Computing technology.

- **Credential Sharing Prevention:** Service and content providers could lose significant revenue if users illegally share access credentials, e.g. for monetary gain. Digital Rights Management (DRM) mechanisms can be used to prevent unauthorised credential sharing.

3 Background

We now introduce the building blocks used to meet the security requirements identified in Section 2. We introduce the Device Management Entity, Manual Authentication (MANA) protocols, the Ninja Service Discovery Protocol, and Trusted Computing technology.

3.1 Device Management Entity (DME)

The *Device Management Entity* (DME) [1] is a semi-distributed, logical construct designed to abstract the technical complexities faced by users (especially non-expert users) in managing (i.e. configuring and operating) their devices in a mobile ubiquitous environment. The DME acts as an interface between user devices and external entities. The DME incorporates the following functional components:

- The **Features Register** maintains information about the capabilities (storage, screen resolution, processing power, battery life, networking, OS, software, etc.) of each device in a user's Personal Distributed Environment. This information assists the DME when deciding which device is most appropriate for use in accessing specific services or content.
- The **Location Register** stores user device location information. It can be used to discover whether devices are reachable or in proximity to the access service.
- The **Security Register** maintains security credentials for users. These could include cryptographic keys, access tokens, and public key certificates.

DMEs are likely to be arranged hierarchically [1], with the root DME at the Network or Service provider, and a local DME in a user's personal area network. Local DMEs and the root DME must communicate securely. Here, for simplicity, we only refer to one instance of the DME, although this could in fact be a collection of DMEs.

In this paper we specify additional security functionality for the security register (of a DME), in order to offer users a richer and more comprehensive set of security services.

3.2 Manual Authentication Protocols

Manual entity authentication (MANA) protocols allow two devices to authenticate one another using a combination of an insecure wireless channel and manual data transfer. MANA protocols are particulary useful in situations where it is necessary to establish a security association (e.g. in the form of a shared key) between two devices in close physical proximity, while minimising the amount of user intervention (e.g. pressing buttons, or reading data from a display). MANA protocols are resistant to Man-in-the-middle (MitM) attacks on the wireless channel. To use a MANA protocol, the devices must possess user interfaces capable of data input (e.g. buttons) and data output (e.g. a display). ISO has standardised four MANA protocols in ISO/IEC 9798-6 [3], designed for use with devices with differing interface capabilities. The properties of the four MANA protocols can be summarised as follows:

- Mechanisms using a short check value:
  - One device with simple input, and one device with simple output;
  - Both devices with simple input capabilities.
- Mechanisms using a Message Authentication Code (MAC):
  - Both devices with simple output capabilities;
  - One device with simple input, and one device with simple output.

There is a growing literature on such mechanisms [2, 4, 5, 7, 8, 11]; we use MANA protocols here because they have been adopted by ISO, and thus these protocols have been subjected to close scrutiny by the security community.

3.3 Ninja Service Discovery Protocol

Service Discovery is an important pre-cursor to service delivery. Many service discovery protocols have been proposed [13]; unfortunately, very few address security and privacy issues (one exception being the scheme of Zhu et al. [14]). It is imperative that service discovery is conducted in a secure and private manner to protect the security and privacy interests of users and service providers.

The Ninja Service Discovery scheme [6] is used to meet the objective of secure service discovery. Ninja offers a number of security features specifically designed for a mobile ubiquitous environment. It provides mutual authentication between user and service provider, preserves the privacy of users, is efficient (has a low communications overhead), and establishes a shared key between the user and service provider. Ninja is therefore well suited to our requirements.

3.4 Trusted Computing

Trusted Computing (TC), as specified by the Trusted Computing Group[1] (TCG) [9, 12], is a technology designed to enhance the security of computing platforms. This aim is met by incorporating a special purpose hardware component, providing so called "roots of trust", into a computing

[1] See http://www.trustedcomputinggroup.org/

platform. This hardware component, known as a Trusted Platform Module (TPM), provides a host platform with a foundation of trust, as well as the basis on which a suite of trusted computing security functionality is built.

The tamper-resistant TPM provides a range of cryptographic functionality (random number generation, RSA signatures, SHA-1, etc.). However, the main distinguishing feature of Trusted Computing is that it allows a platform to attest to its state to a remote verifier, thereby providing users with assurance that the platform with which they are interacting is behaving in the expected manner, and has not been compromised. Attestation is achieved using a process known as Integrity Measurement, Storage and Reporting (IMSR), as follows.

A platform's operational state is measured using the Root of Trust for Measurement (RTM). State measurements, also known as integrity metrics, are recorded to a file called the Stored Measurement Log (SML). At the same time, the Root of Trust of Storage (RTS) computes a digest (i.e. a cryptographic hash) of the integrity metrics, which is written into one of the TPM's internal Platform Configuration Registers (PCRs). These PCRs are 160 bits long (the SHA-1 output length). Since the TPM only has a limited number of PCRs, and in order to preserve the integrity and order of previous measured events (potentially a very large number), TPMs use a technique known as *extending the digest*. Newly measured values are appended to the current contents of a PCR, rehashed and then stored back in the relevant PCR. Finally, the Root of Trust of Reporting (RTR) provides a challenger with the requested integrity metrics (i.e. the SML and the corresponding PCR values). To prove the integrity metrics originate from a TPM, the PCR values are signed using one of the TPM's Attestation Identity Keys (AIKs). This completes the IMSR process.

Another important feature of Trusted Computing used in the SDMF framework is the *Protected Message Exchange Mechanism*, namely *Binding* and *Sealing*. Binding involves encrypting data with a non-migratable public key, i.e. a public key stored by the TPM and for which the private key will never leave the TPM. As a result, bound data is available only to the TPM. Sealing involves binding data to integrity metrics representing a particular platform state. The TPM will only make the data available to the platform if the current values in the PCRs are equal to those bound to the data, i.e. only if the platform is in the pre-specified state. Note that arbitrary data as well as keys can be bound or sealed to a TPM and to particular platform states.

4 A Framework for Secure Device Management

We now present the Secure Device Management Framework (SDMF), which is designed to secure the process of service provisioning to end user devices. At the heart of this framework is the Device Management Entity (DME), that manages a user device's security credentials on its behalf, and interacts with service providers. This framework also provides users with assurance that a compromised device is unable to consume the delivered service, and, at the same time, prevents users from illegally sharing their credentials with other users. We achieve these security objectives using Trusted Computing functionality.

We first introduce the entities in the SDMF framework. We then state the assumptions upon which this framework is based. Finally, we describe the operation of the framework.

4.1 The Entities

The entities participating in the Secure Device Management Framework are as follows:

- The **User**, often a human, who is the end consumer of a service or application.
- The **User Devices** (such as PDAs, laptops, smartphones) which are used by the user to access or consume the available services.
- The **Device Management Entity** (**DME**) performs a number of functions (as discussed in Section 3.1). In the framework, its main role is to securely interact with service providers on behalf of user devices, and to manage the security credentials of user devices.
- The **Service Provider** provides end users with a range of service offerings (e.g. music, videos, software applications, etc.).

4.2 The Working Assumptions

The SDMF relies on a number of assumptions:

- User devices are equipped with Trusted Computing functionality conforming to version 1.2 of the TCG TPM specifications [12].
- User devices have a (possibly very simple) interface capable of data input and data output.
- The DME is trusted by both the user and service provider to perform its tasks correctly.
- The user has ownership control of the TPM in the device. The user therefore possesses an authorisation secret that can later be used to authenticate to the TPM.
- User devices and the DME have loosely synchronised clocks. This enables the user devices and the DME to check that a received message is fresh.

- Prior to the enrolment phase, the TPM in the User Device has generated an Attestation Identity Key pair (AIK_{pk}, AIK_{sk}), and has obtained a Certificate $Cert_{AIK}$ for AIK_{pk} from a Privacy CA (a trusted computing specific trusted third party).

Finally note that our framework is independent of the underlying transport and network layer protocols, as it is purely an application layer solution.

4.3 The SDMF

The notation used is summarised in Table 1.

Table 1. Notation

Notation	Description
D	The User Device
M	The Device Management Entity (DME)
S	The Service Provider
$EnReq$	Enrolment Request Message
$EnRep$	Enrolment Reply Message
$EnMsg$	Enrolment Message
$EnAck$	Enrolment Acknowledgement Message
$DevID$	A Unique Device ID number
$SessID$	A Unique Session ID number
$\|\|$	The concatenation operator
t_A	A timestamp generated by principal A
N	A nonce (a random value)
ID_A	The identity of principal A
(PK_A, SK_A)	The public and private Key pair of principal A
H	A cryptographic hash-function
$E_K(M)$	The encryption of a message, M, using the key K
$D_K(M)$	The decryption of a message, M, using the key K
$\text{MAC}_K(M)$	The message authentication code (MAC) of a message, M, computed using the key K
$Sig_K(M)$	A signature on a message, M, signed using the key K

As shown in Figure 2, the SDMF involves five phases. We now describe the workings of each phase in greater detail.

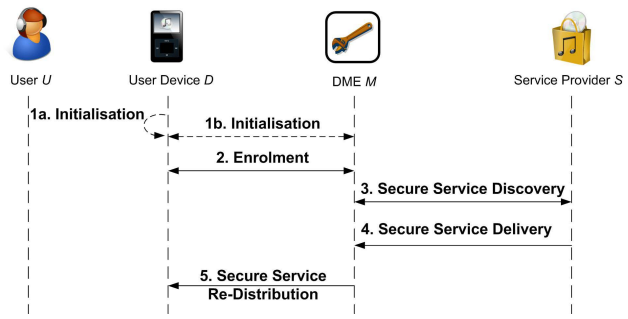

Figure 2. The Secure Device Management Framework Phases of Operation

Initialisation Phase. This phase involves the user preparing a device for the subsequent phases. One of the primary aims of this phase is for the user device and the DME to establish a shared key, used to secure subsequent communications. The user U, the user device D, and the DME M must be in close physical proximity. The user performs the following steps:

1. The user authenticates himself/herself to the TPM in the device D using the authorisation secret (obtained earlier when the user took ownership of the TPM). This enables the user to access the TPM functionality.
2. The user instructs D and the DME M to engage in a MANA protocol (see Section 3.2). The choice of protocol will depend on the types of interface (input and display) that D possesses. The User must take part in the protocol and perform his/her role in the protocol correctly (e.g. reading data from the display accurately and pressing the correct buttons). At the end of the protocol execution, D and M will have securely established a shared secret key K_{DM}. This is a long lived master key used mainly to established subsequent session keys, and should be stored securely when not in use.
3. The DME sends a copy of its public key PK_M to the user device as follows:

$$M \rightarrow D : (PK_M, \text{MAC}_{K_{DM}}(PK_M)).$$

The user repeats the above procedure for every device he/she owns (or intends to use for service consumption).

Enrolment Phase. In this phase, device D is securely enrolled with DME M, which involves M being given D's security credentials. This phase may take place at any time after the Initialisation phase, and does not require D and M to be in close physical proximity. If necessary this phase can be performed at regular intervals, or even for every 'session', without repeating the Initialisation phase. It involves the following steps.

1. D constructs an Enrolment Request message, $EnReq$, which includes a Device Identifier $DevID$, a Session Identifier $SessID$, and a timestamp t_D:

$$EnReq = (DevID, SessID, t_D).$$

Using K_{DM} (established in the Initialisation Phase), D derives session keys $K_1 = H(0||K_{DM}||SessID)$ and $K_2 = H(1||K_{DM}||SessID)$, where H is a cryptographic hash function (e.g. SHA-1 [10]). Note that K_1 and K_2 will be discarded at the end of the enrolment phase, and must not be reused. D then computes a MAC on $EnReq$ using K_2, and sends $EnReq$ and the MAC to M:

$$D \rightarrow M : (EnReq, \text{MAC}_{K_2}(EnReq)).$$

2. On receiving the above message, M retrieves $SessID$, and computes K_1 and K_2 in the same way as D. M then verifies the integrity of the received message using K_2, and (if the outcome is satisfactory) prepares an Enrolment Reply message $EnRep$ for D:

$$EnRep = (ID_M, DevID, SessID, n, t_M),$$

where n is a nonce. M computes a MAC on $EnRep$ using K_2, and then sends $EnRep$ and the MAC to D:

$$M \rightarrow D : (EnRep, \text{MAC}_{K_2}(EnRep)).$$

3. D checks the integrity of the above message and, if it verifies correctly, then reports its "trustworthiness" to M by instructing its TPM to attest to its current software state. This involves D's TPM signing the current values of the PCRs requested by M, denoted by PCR where the PCR values are concatenated with the nonce n supplied by M. Note that this signature is sent in conjunction with the portions of the SML necessary to interpret and verify the PCR values:

$$Sig_{AIK_{sk}}(PCR||n).$$

Note that the nonce n is included to prevent possible replay attacks.

4. The TPM in D generates a non-migratable Service Key Encrypting Key pair $(SKEK_{pk}, SKEK_{sk})$ which is bound to the TPM in D, and sealed to D's current software state (i.e. the PCR values used in the previous step) using the `TPM_CreateWrapKey` command. The TPM then uses AIK_{pk} to sign a certificate $Cert_{SKEK_{pk}}$ for $SKEK_{pk}$ using the `TPM_CertifyKey` command. $Cert_{SKEK_{pk}}$ contains a digest of $SKEK_{pk}$, and also describes the properties of the key, including the key type (bind, identity, signing, or storage), whether the key is migratable, and the PCRs to which the key is sealed. M will later use $SKEK_{pk}$ to encrypt services destined for this device.

5. D constructs an Enrolment Message $EnMsg$:

$$EnMsg = (DevID, SKEK_{pk}, Cert_{SKEK_{pk}}, t_D, SML, Sig_{AIK_{sk}}(PCR||n), AIK_{pk}, Cert_{AIK}).$$

D then encrypts $EnMsg$ using K_1, and computes a MAC on the encrypted $EnMsg$ using K_2 to give:

$$E_{K_1}(EnMsg) \quad \text{and} \quad \text{MAC}_{K_2}(E_{K_1}(EnMsg)).$$

D sends the following to M:

$$D \rightarrow M : (DevID, E_{K_1}(EnMsg), \text{MAC}_{K_2}(E_{K_1}(EnMsg))).$$

6. On receiving the above message, M checks its integrity using K_2, and then decrypts the encrypted $EnMsg$ using K_1. M next verifies the signature on the PCR values using AIK_{pk}, and assesses the "trustworthiness" of D by comparing the reported integrity metrics (i.e. the PCR values and the accompanying SML) against a list of known trustworthy states. If the device is deemed to be in a trustworthy state, M creates a new entry for this device in its security register. The security register holds information associated with this device, i.e. $DevID$, K_{DM}, AIK_{pk}, and $SKEK_{pk}$. At this point, the user device D is successfully enrolled with the DME M.

7. M creates an Enrolment Acknowledgement message, $EnAck$ to inform D of the enrollment outcome $EnOut$ (i.e. Success or Failure):

$$EnAck = (ID_M, DevID, EnOut, t_M).$$

M computes a MAC on $EnAck$ using K_2, and sends the following to D:

$$D \rightarrow M : (EnAck, \text{MAC}_{K_2}(EnAck)).$$

At any subsequent time, the User Device D must be in the state indicated by the values of its PCRs in order to be able to access services via the DME.

Secure Service Discovery Phase. In this phase, the Ninja protocol (see Section 3.3) is used to secure the process of service discovery between the DME (acting on behalf of the user devices) and a service provider. Prospective service providers advertise their service offerings via authenticated service advertisement messages. If a user is interested in a particular advertised service, the DME M can respond with a service reply message on behalf of the user devices. Service interactions between the DME and service provider are secured. At the end of the Ninja service discovery protocol run, a shared secret key K_{MS} has been established between the DME and the service provider. This shared key is used to secure the delivery of the sought services in the Secure Service Delivery phase, described immediately below.

Secure Service Delivery Phase. In this phase, the service provider securely delivers a requested service, Srv, to the DME.

1. The service provider generates a secret service encryption key, K_{Srv}, and uses it to encrypt the service:

$$E_{K_{Srv}}(Srv).$$

The service provider encrypts K_{Srv} using K_3 to obtain $E_{K_3}(K_{Srv})$, and computes a MAC on the encrypted

service and the encrypted copy of K_{Srv} using K_4, where $K_{MS} = K_3||K_4$, to obtain:

$$\text{MAC}_{K_4}(E_{K_{Srv}}(Srv)||E_{K_3}(K_{Srv})||t_S).$$

2. The service provider sends the following to the DME:

$$S \rightarrow M : (E_{K_{Srv}}(Srv), E_{K_3}(K_{Srv}), \text{MAC}_{K_4}(E_{K_{Srv}}(Srv)||E_{K_3}(K_{Srv})), t_S).$$

3. On receiving the above message, the DME checks its integrity by recomputing the MAC using K_4, and then comparing it with the received value. If the two values agree, the DME proceeds to the next step.

4. The DME decrypts $E_{K_3}(K_{Srv})$ using K_3 to obtain K_{Srv}. Key K_{Srv} will be securely stored by the DME.

Secure Service Re-Distribution Phase. In this phase, the DME M securely redistributes the service (received from the service provider in the Secure Service Discovery phase) to a specific Device D.

1. M consults its security register, retrieves the corresponding device public key $SKEK_{pk}$ (which was sealed to a trustworthy platform state), and re-encrypts K_{Srv} using $SKEK_{pk}$:

$$E_{SKEK_{pk}}(K_{Srv}).$$

2. M generates a new *SessID* and uses it to derive a new session key $K_5 = H(DevID||K_{DM}||SessID)$. M then computes a MAC on the encrypted service and the re-encrypted service key using K_5 to obtain:

$$\text{MAC}_{K_5}(E_{K_{Srv}}(Srv)||E_{SKEK_{pk}}(K_{Srv})).$$

3. M sends the secured service (i.e. the encrypted service) and the MAC to D:

$$M \rightarrow D : (SessID, E_{K_{Srv}}(Srv), E_{SKEK_{pk}}(K_{Srv}), \text{MAC}_{K_5}(E_{K_{Srv}}(Srv)||E_{SKEK_{pk}}(K_{Srv}))).$$

To consume the service, the user device performs the following steps:

1. On receiving the above message, the user device first computes K_5, and then uses it to check the integrity of the received message.

2. If the outcome of the previous step is satisfactory, the user device unseals $SKEK_{sk}$ using the `TPM_Unseal` command. Note that this step is only possible if the TPM in the device is in the same state as it was when the key was created and sealed.

3. If the previous step is successful, the user device decrypts $E_{SKEK_{pk}}(K_{Srv})$ using $SKEK_{sk}$ to obtain K_{Srv}.

4. The User Device decrypts $E_{K_{Srv}}(Srv)$ using K_{Srv} to obtain the service Srv.

The service can now be consumed on device D.

5 Security Analysis and Discussion

We now evaluate SDMF against the security requirements identified in Section 2.2.

Entity Authentication: Entity authentication takes place between the following pairs of entities:

- User-to-Device: During the initialisation phase, the user is authenticated by the TPM in the device using an authorisation secret. Only the user (i.e. the authorised custodian of the TPM and the device) knows this secret. An attacker who has stolen the device would therefore be unable to access the TPM functionality required to unlock and access a service.

- Device-to-DME: During the enrolment and the secure service re-distribution phases, a secret key known only to the device and DME (i.e. K_{DM} and keys derived from it) is used to compute MACs on the service messages, providing device/DME mutual authentication.

- DME-to-Service Provider: The DME and service provider are mutually authenticated in the service discovery phase.

Timestamps and nonces are included in the service messages to address replay attacks.

Integrity Protection: All message exchanged during the various phases are integrity protected.

In the Enrolment and Secure Service Re-Distribution phases, the integrity of the $EnReq$, $EnRep$, $EnMsg$, and $EnAck$ messages, the Service Srv, and the encrypted service key K_{Srv}, are protected with MACs that are computed using a shared key known only to the device and the DME (i.e. K_2 and K_5).

In the Secure Service Delivery phase, the Service Srv and the service encryption key K_{Srv} are integrity protected with MACs computed using a secret key known only to the DME and the service provider. Data origin authentication is also achieved for the service that is delivered to the DME by the service provider.

Service Confidentiality: In the Enrolment Phase, the Enrolment message $EnMsg$ is encrypted before it is sent to the DME. This prevents eavesdroppers reading it. In the Service Delivery and Service Re-distribution phases, the

service, Srv, as well as the service encryption key, K_{Srv}, are encrypted whilst in transit. Eavesdroppers thus cannot learn the type of services a user is consuming by observing the messages. Also, active attackers cannot capture the encrypted service and consume it.

Device Integrity Assurance: If a user device has been compromised (e.g. by malicious software), its software (as recorded in the PCR values) will differ from the expected value, and hence the device will be prevented from accessing a service, since it will now be unable to perform the UnSeal operation to retrieve the necessary secret key. If the device is unable to access a service, a user will be able to deduce that the device may have been compromised. Since a compromised (or untrustworthy) device is "automatically" prevented from accessing a service, the requirement for a revocation infrastructure (for distributing and maintaining up to date CRLs) may no longer be necessary.

Unauthorised Credential Sharing Prevention: Depending on the service plans a user subscribes to, service providers may, for example, allow a user to access a service on a maximum number (three, say) of personal devices. These three devices are enrolled with the DME. As a service is encrypted with an service encryption key, and the service encryption key is encrypted using a key which is bound to the device's TPM, a service can thus only be consumed on the device to which the key is bound. The private part of the Service Key Encrypting Key $SKEK_{sk}$ never leaves the platform, and so a user cannot share it. Even the key is extracted, it may not be possible to unseal it in another device, as the PCR values will probably differ.

6 Conclusion

We have introduced SDMF, a framework for the secure delivery of ubiquitous services in a mobile environment. Using Trusted Computing and certain other security mechanisms, the framework facilitates the secure delivery of services to user devices, whilst removing the complexities of security management from the users. A user need not be worried if his/her device has been compromised. Service providers are also assured that unauthorised credential sharing is prevented. Our security analysis shows that the framework meets all the identified security objectives.

Acknowledgment

The work reported in this paper has formed part of the Ubiquitous Services Core Research Programme of the Virtual Centre of Excellence in Mobile & Personal Communications, Mobile VCE, www.mobilevce.com.

References

[1] R. C. Atkinson, J. Irvine, J. Dunlop, and S. Vadagama. The personal distributed environment. *IEEE Wireless Communications*, 14(2):62–69, 2007.

[2] D. Balfanz, D. K. Smetters, P. Stewart, and H. C. Wong. Talking to strangers: Authentication in ad hoc wireless networks. In *Proc. of Network and Distributed Systems Security Symposium 2002 (NDSS'02)*. The Internet Society, Reston, Virgina, 2002.

[3] International Organization for Standardization. ISO/IEC 9798–6, Information technology — Security techniques — Entity authentication — Part 6: Mechanisms using manual data transfer, 2005.

[4] T. Kindberg and K. Zhang. Secure spontaneous device association. In A. Dey et al., editors, *5th International Conference on Ubiquitous Computing (Ubicomp'03)*, pages 124–131. Springer-Verlag LNCS 2864, Berlin, 2003.

[5] S. Laur and K. Nyberg. Efficient mutual data authentication using manually authenticated strings. In D. Pointcheval et al., editors, *5th International Conference on Cryptology and Network Security (CANS 2006)*, pages 90–107. Springer-Verlag LNCS 4301, Berlin, 2006.

[6] A. Leung and C. J. Mitchell. Ninja: Non identity based, privacy preserving authentication for ubiquitous environments. In J. Krumm et al., editors, *9th International Conference on Ubiquitous Computing (Ubicomp 2007)*, pages 73–90. Springer-Verlag LNCS 4717, Berlin, 2007.

[7] R. Mayrhofer and H. Gellersen. Shake well before use: Authentication based on accelerometer data. In A. LaMarca et al., editors, *5th International Conference on Pervasive Computing (Pervasive 2007)*, pages 144–161. Springer-Verlag LNCS 4480, Berlin, 2007.

[8] J. M. McCune, A. Perrig, and M. K. Reiter. Seeing-is-believing: Using camera phones for human-verifiable authentication. In *Proc. of the 2005 IEEE Symposium of Security and Privacy (SP'05)*, pages 110–124. IEEE Computer Society, 2005.

[9] C. J. Mitchell, editor. *Trusted Computing*. IEE Press, London, 2005.

[10] National Institute of Standards and Technology (NIST). Secure Hash Standard. Federal information processing standards (FIPS) publication 180-2, 2002.

[11] C. Soriente, G. Tsudik, and E. Uzun. BEDA: Button-enabled device association. In *Proc. of UbiComp 2007 Workshops: First International Workshop on Security for Spontaneous Interaction (IWSSI 2007).*, pages 443–449, 2007.

[12] Trusted Computing Group (TCG). TCG Specification Architecture Overview. Version 1.2, The Trusted Computing Group, Portland, Oregon, USA, April 2004.

[13] F. Zhu, M. Mutka, and L. Li. Service discovery in pervasive computing environments. *IEEE Pervasive Computing*, 4(4):81–90, 2005.

[14] F. Zhu, M. Mutka, and L. Ni. A private, secure and user-centric information exposure model for service discovery protocols. *IEEE Trans. on Mobile Computing*, 5(4):418–429, 2006.