Exercise 5: Attestation

1 Introduction

In this exercise we will introduce the Integrity Reporting mechanism *attestation*, mainly attestation with Privacy CA and Direct Anonymous Attestation (DAA). In the practical exercises, we will learn how to implement a simplified attestation protocol with the TCG Software Stack (TSS)[5]. The practical exercise highly depends on exercise 2 "Simple Programming - Extending a PCR" and exercise 4 "Data Sealing", so please review them before you start with this one.

Attestation is an ability of the TPM to show that the host system configuration is authentic to a remote verifier . In order to ensure that, the TPM and Core Root of Trust for Measurement (CRTM) (also see exercise 3 "Authenticated Boot") act as Root of Trust for Reporting (RTR).

1.1 Root of Trust for Reporting

In TCG systems, there are three Roots of Trust used to describe the platform characteristics that affect the trustworthiness of the platform[1]: Root of Trust For Measurement (RTM), Root of Trust for Storage (RTS) and Root of Trust for Reporting (RTR). The RTR is a piece of code capable of reporting integrity measurements and vouching for the authenticity of PCR values based on trusted platform identities using an Attestation Identity Key (AIK). Here integrity measurements are digitally signed to authenticate PCR values.

1.2 Attestation Identity Key (AIK)

Intuitively, an AIK is an alias for the platform-unique key, the Endorsement Key (EK) [2]. The EK cannot perform signatures for security reasons and due to privacy concerns. If a digital signature was performed by the EK, then any entity could track the use of the EK. So the use of the EK as a signature is cryptographically sound, but this does not ensure privacy.

The TPM can create an unlimited number of AIKs. The AIK is an asymmetric key pair only used for signing, and is never used for encryption. It only signs information generated internally by the TPM, e.g., PCR values. The AIK must never sign arbitrary external data, since it would be possible for an attacker to create a block of data that appears to be a PCR value.

1.3 AIK credential

The AIK credential identifies the AIK private key used to sign PCR values. By issuing the AIK credential, the issuer attests the authenticity of the TPM. For this, the host platform must prove to the issuer that the TPM owns the AIK and the AIK is tied to valid endorsement, platform and conformance credentials[1]. A verifier could use this

information, along with other information in the credential to trust the platform via an attestation protocol.

1.4 Attestation with Privacy CA

One approach to attestation relies on a trusted certificate authority, called Privacy CA, which issues AIK credentials. The Privacy CA is trusted not to reveal sensitive information. It is also trusted not to misrepresent the trust properties of platforms for which AIK credentials are issued. For instance, a Privacy CA is trusted not to issue AIK credentials when the verification of the endorsement credential fails.

1.4.1 AIK Creation with Privacy CA

AIK creation involves both TPM and Privacy CA, which acts as a Trusted Third Party (TTP).

1. The `TPM_MakeIdentity` command[4] causes the TPM to generate the AIK key pair.
2. The TPM enrolls AIK public keys with a Privacy CA. Enrollment with a Privacy CA requires the TPM to prove AIK keys are exclusively bound to the TPM. The platform accomplishes this by decrypting the AIK credential using the EK private key in the TPM. Only the TPM with the EK private key will be able to perform the decryption. CA must also check endorsement, platform and conformance credentials.
3. The Privacy CA then distributes a credential certifying the AIK.
4. `The TPM_ActivateIdentity` command[4] unwraps a session key that allows for the decryption of the AIK credential. The session key was encrypted using the public EK and requires the private EK to perform the decryption, hence only the TPM can decrypt the AIK credential.

1.4.2 Attestation Protocol

Integrity reporting may be used to determine a platform's current configuration. A protocol for reporting integrity measurements is illustrated in Figure 1.

The attestation protocol consists of several steps:

1. A verifier requests one or more PCR values and other relevant information (e.g. TPM version) from a platform, and sends a nonce[1] to this platform.
2. An agent on the platform containing a TPM requests the TPM to sign the PCRs, nonce and attestation interests (e.g., whether the TPM version info has to be included) with the AIK, using the `TPM_Quote`/`TPM_Quote2` command[4].
3. The TPM signs requested data.

[1] An unpredictable random value used during a cryptographic operation to prevent replay attack.

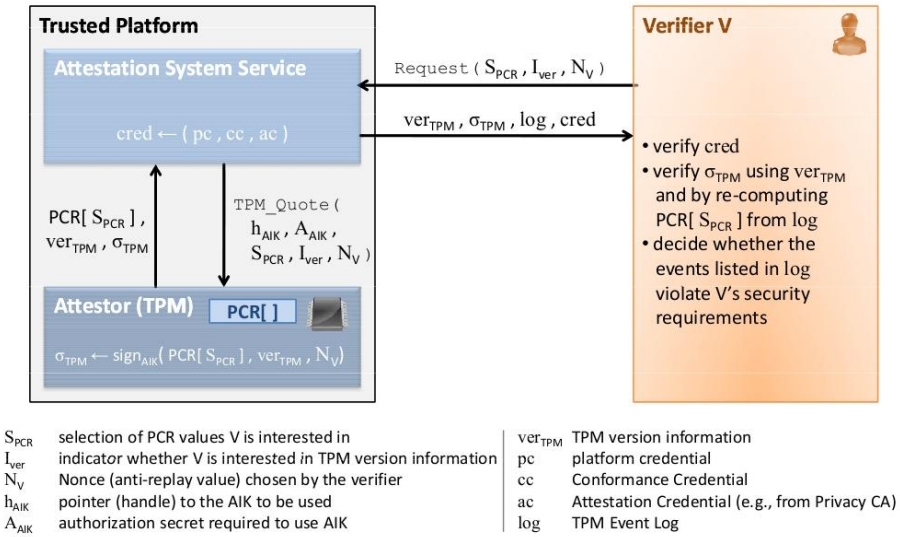

Figure 1: Attestation after a Privacy CA has issued AIK credential

4. The agent receives PCR values from the TPM, collects eventlogs, other concerned information, and credentials that vouch for the TPM.

5. The agent receives signed data from TPM and sends it together with above data to the verifier.

6. The verifier verifies the signed data. The measurement digest is computed and compared with PCR value. The AIK credential is evaluated and signatures are checked.

1.4.3 Structures related to Attestation Implementation

`Tspi_TPM_Quote/Tspi_TPM_Quote2` uses the `TSS_VALIDATION` structure[5], which provides the ability to verify signatures and validation digests built over certain TPM command parameters. These parameters are returned as a byte stream and are defined within [2]. The caller usually provides some random data (external Data value) as input, which is included in the signature/digest calculation. The structure is shown below:

```
1 typedef struct tdTSS_VALIDATION
2 {
3     TSS_VERSION    versionInfo;
4     UINT32         ulExternalDataLength;
5     BYTE*          rgbExternalData;
6     UINT32         ulDataLength;
7     BYTE*          rgbData;
```

```
8     UINT32          ulValidationLength;
9     BYTE*           rgbValdationData;
10 } TSS_VALIDATION;
```

1.5 Direct Anonymous Attestation (DAA)

Although AIK creation with Privacy CA can achieve the necessary privacy characteristics, it still requires trust in the Privacy CA. Moreover, every time the TPM generates an AIK, it needs to request Privacy CA to issue corresponding AIK certificates, which could make the Privacy CA a bottleneck when serving a massive number of TPMs. The new Direct Anonymous Attestation (DAA)[7] method was adopted by the TCG for AIK creation as a means to achieve enhanced privacy by eliminating the need to trust the TTP for privacy in the attestation protocol. The TTP that replaces the Privacy CA is called DAA Issuer.

DAA is based on a family of advanced cryptographic techniques known as Zero Knowledge Proofs (ZKP). From a privacy perspective, the advantage of DAA is that even the Issuer cannot link an attestation to an EK (and thus to a particular TPM). Moreover, nobody, not even the Issuer, can link two attestations (DAA signatures), i.e., it is impossible to distinguish if they are coming from the same TPM or not.

From a scalability perspective, the advantage of DAA is that a TPM can create and certify an unlimited number of AIKs, given only one interaction with the DAA Issuer.

DAA consists of two protocols: Join and Sign. During Join (Figure 2), the TPM (under TPM Owner control) interacts with an Issuer to generate a set of DAA credentials. This can be done multiple times. In this protocol, the Issuer must also verify the endorsement, platform, and conformance credentials. A secure channel between TPM and Issuer has to be established (based on the EK) in order to prevent an attacker from "simulating" the TPM.

The DAA credentials are used during an interaction defined by the DAA Sign protocol (see Figure 3) with a second party called the Verifier. The goal of the protocol is that the Verifier can determine if the TPM contains a valid set of DAA credentials from a particular Issuer, but does not have specific knowledge that might help to identify the TPM from among others that also have valid DAA credentials from the same Issuer. The DAA Sign protocol can also be used to sign AIKs. A Verifier can then check the signature, which confirms that this is a valid AIK from a TPM with DAA credentials from the given Issuer. Therefore a TPM can create and sign an arbitrary number of AIKs based on one set of DAA credentials obtained in a single execution of the Join protocol.

1.6 Hash Objects

Hash objects hold hash values, can compute the hash of data, and can sign and verify hashes using keys. Natively, the TSS supports hashing data using the SHA1 algorithm only, although it can sign and verify hashes of any type. To create a SHA1 hash of some data, use `Tspi_Hash_UpdateHashValue`.

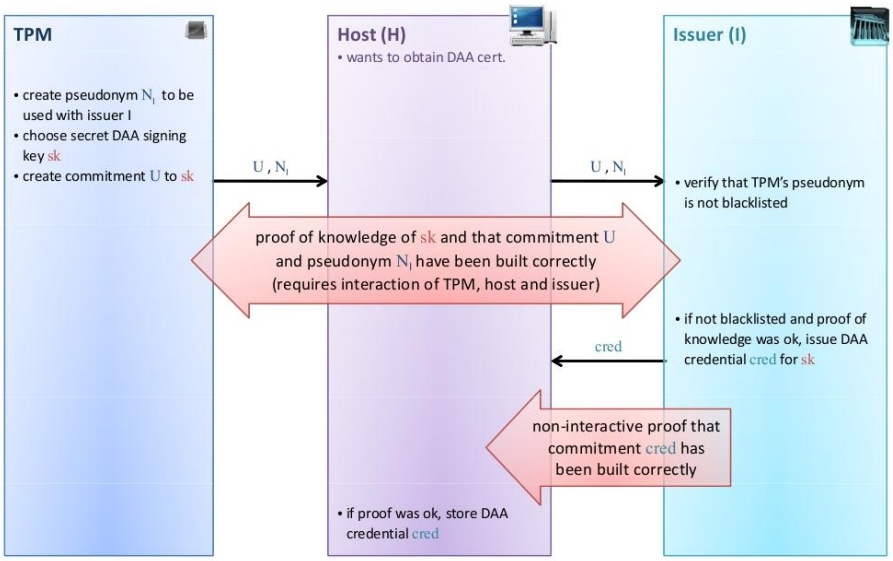

Figure 2: Direct Anonymous Attestation: Join Protocol

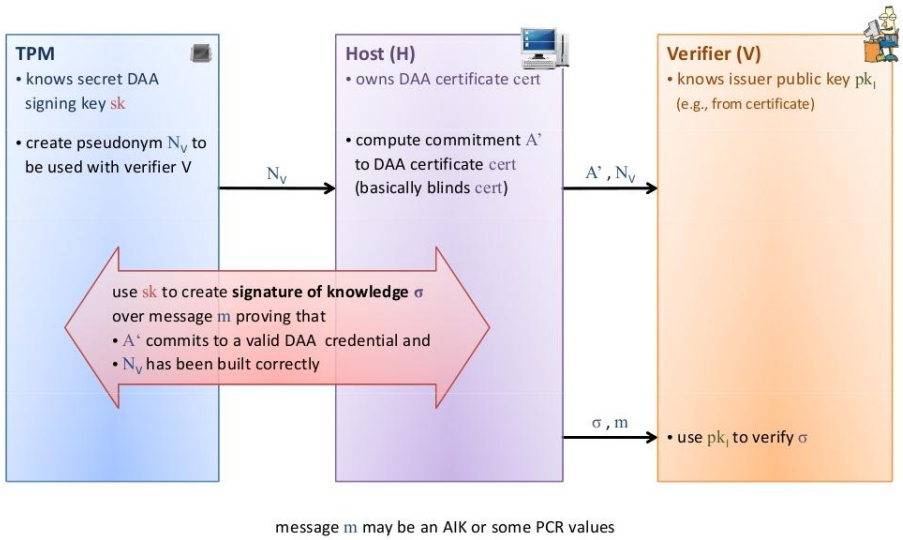

Figure 3: Direct Anonymous Attestation: Sign Protocol

2 Theoretical Assignments

1. What is an RTR? If the RTR of a Web Server is tampered or malfunctions, could you still trust it? Justify your answer!

2. Suppose you are an online retailer, selling and buying products using the same computer. Should you use the EK or one AIK for both occasions? Justify your answer!

3. What's the role of the Privacy CA in "Attestation with Privacy CA"? If a Privacy CA is compromised, what damages can it possibly bring about? Name At least two.

4. Why does the attestation protocol with Privacy CA use a nonce?

5. Suppose you use attestation with Privacy CA to attest a remote business partner's platform. Could you trust that the platform you attest is the platform you're communicating with? Justify your answer!

6. What benefits can be achieved by using DAA instead of using Attestation with Privacy CA? Name at least two advantages.

3 Practical Assignments

3.1 AIK Certificate Creation

Your task is retrieving an AIK certificate to understand the AIK creation process with Privacy CA.

Privacy CA[9] defines an API designed as a straightforward interface to the TCG-defined Privacy CA functionality. It provides a Privacy CA server and corresponding client functions to retrieve AIK certificates.
The client sample code can be downloaded here: `http://privacyca.com/identity.c`. For the purpose of this exercise, a modified version will be used, which is located in *Exercise Data*.

3.1.1 Request and retrieve AIK Certificate from Privacy CA

In this part, you need to compile the sample code and get the AIK certificate. Compile it with `gcc identity.c -o identity -ltspi -lcurl` and run the program. The program creates a new AIK and receives a certificate for it. Further, the program writes the key to a file for later use by `attest.c`. Is it secure to save the key to the disk? Justify your answer!

3.1.2 Verify the AIK Certificate

In this part, you need to download the Privacy CA Root Certificate and Privacy CA Level 0 Certificate from the Privacy CA website. Save both of them to one file. Use the `openssl` tools to verify the AIK certificate received from the Privacy CA in the above exercise. Write down the command you used.

Hint: See `openssl` manpage on "verify".

3.2 Implementing A Simplified Attestation Protocol on your own and Verifying it

Your task is to extend two given programs to implement a simplified attestation protocol. The programs contain the attestation and verification process on host and verifier's platform separately. The attestation program is extended based on the sample code you used in the last exercise.

In the simplified attestation protocol, Privacy CA doesn't actually check EK/platform certificates and just issues the AIK certificate for the client. And in order to reduce the programming complexity, the programs don't communicate via the network.

The attestation protocol programs perform the task as follows:

- Attestation process: The host creates an AIK and gets an AIK certificate for later Quote (see sample code).
- Attestation process: The host selects some PCRs extended by the BIOS and TrustedGrub: PCR[0], PCR[4], PCR[5], PCR[8], PCR[9], PCR[12], PCR[13], PCR[14], (see also exercise 3). The verifier is supposed to know the reference PCR values for later comparison.
- Attestation process: The host quotes the selected PCR values with a nonce (usually provided by the verifier; you can put any data here for this example).
- Attestation process: The host writes the quoted data and the public key of the AIK into an attestation file on the host platform.
- Verification process: The verifier reads the quoted data and the public key of the AIK from the attestation file on verifier's platform.
- Verification process: The verifier verifies the attestation file with the public key of the AIK and compares the PCR values to reference PCR values.

3.2.1 Selecting Attested PCR on Host Platform

In this part, you need to select the PCRs which are sent to the host by the verifier. Extend `attest.c` according:

1. Create a PCR object (Hint: Appendix, Table 1).
2. Select PCRs 0, 4, 5, 8, 9, 12, 13 and 14.

3.2.2 Quoting PCR on Host Platform

In this part, you need to quote selected PCRs on the host platform:

1. Create a TPM object (Hint: Refer to Exercise 4).
2. Use `Tspi_TPM_Quote` to sign the set of the selected PCRs.

3.2.3 Writing Quoted Data and Public Key to Attestation File on Host Platform

In this part, you need to write the quoted data and public key of the signing key to a file on the host platform. There is already a function implemented, called `writeCert`, which writes the public key and the validation structure to a buffer, which is written to a file later.

1. Get the public key of the signing key using `Tspi_Key_GetPubKey`.
2. Write the result of `Tspi_TPM_Quote`, which is a `TSS_VALIDATION` structure, together with the obtained public key to an "attestation file". Use the provided function `writeCert`.

3.2.4 Reading Attestation File on Verifier's Platform

There is a function implemented for this section too - `readCert` does the exact opposite operations as the function `writeCert` mentioned above.

1. Copy the attestation file from host platform to verifier platform.
2. Read the attestation file to get the public key of signing key and the signature (Hint: Use the provided function `readCert`).
3. You will find a copy of the *referencepcrs* file in *Exercise Data*. Check if the values in the file match those on the host's platform and if not write the correct values.

3.2.5 Verify Attestation File on Verifier's Platform

Now you need to verify the attestation file and reference PCR values on the verifier's platform. To reduce the complexity, the last function you will need is also provided - `readPCRFile` will extract the predefined PCR values, together with the PCR indices from the *referencepcrs* file. You need those in order to calculate a composite hash, which is compared (in the last step of this exercise) to the hash that you received.

1. Create a hash object (Hint: Appendix, Table 1).
2. Use `Tspi_Hash_UpdateHashValue` to update it with `rgbData` from the `TSS_VALIDATION` structure.
3. Create an empty RSA Key Object.
4. Use `Tspi_SetAttribUint32` (see [5, p.60 "Attribute Definitions for a Key Object"] and [5, p.89 "Key Signature Scheme Definitions"]) and `Tspi_SetAttribData` [5, p.60] to update the newly created Key Object with the public key from the attestation file. As you will be verifying a signature, think of the flags that you have to use.
5. Use `Tspi_Hash_VerifySignature` to verify the signature from `TSS_VALIDATION`.
6. Call the function `readPCRFile` to read the PCR values from the *referencepcrs* file.
7. Call the function `Compute_PCRComposite_Hash` with the reference PCR values and compare the result with the PCR data from `TSS_VALIDATION`. The PCR Data starts from Byte 8 of `v.rgbData` and has the same length as `PCRCompositeHash`.

3.2.6 Modifying and Testing

Modify the PCR values on the host platform by extending selected PCR values. You should be able to do it as in exercise 2 "Simple Programming - Extending a PCR Register". To extend a PCR use the program provided in the *Exercise Data* folder or use your own, developed in Exercise 2.

Now run Attestation and Verification programs again, and explain what happens.

Appendix

Object Type	Description
TSS_OBJECT_TYPE_POLICY	Policy object
TSS_OBJECT_TYPE_RSAKEY	RSAKey object
TSS_OBJECT_TYPE_ENCDATA	Encrypted data object; sealed data or bound data
TSS_OBJECT_TYPE_PCRS	PCR composite object
TSS_OBJECT_TYPE_HASH	Hash object
TSS_OBJECT_TYPE_NV	Non Volatile RAM object
TSS_OBJECT_TYPE_MIGDATA	CMK-Migration data object
TSS_OBJECT_TYPE_DAA	DAA object

Table 1: Object Types for `Tspi_Context_CreateObject`

References

[1] TCG Architecture Overview Version 1.4,
http://www.trustedcomputinggroup.org/files/resource_files/AC652DE1-1D09-3519-ADA026A0C05CFAC2/TCG_1_4_Architecture_Overview.pdf

[2] TPM Specification Version 1.2 Revision 103: Part 1 - Design Principles,
http://www.trustedcomputinggroup.org/

[3] TPM SpecificationVersion 1.2 Revision 103: Part 2 - Structures,
http://www.trustedcomputinggroup.org/

[4] TPM SpecificationVersion 1.2 Revision 103: Part 3 - Commands,
http://www.trustedcomputinggroup.org/

[5] TCG Software Stack (TSS) Specification Version 1.2,
http://www.trustedcomputinggroup.org/files/resource_files/6479CD77-1D09-3519-AD89EAD1BC8C97F0/TSS_1_2_Errata_A-final.pdf

[6] A Practical Guide to Trusted Computing, D. Challener, K. Yoder, R. Catherman, D. Safford and L. Van Doorn, 2008, ISBN-10: 0132398427

[7] Direct anonymous attestation, Ernie Brickell, Jan Camenisch and Liqun Chen, 2004, CCS '04: Proceedings of the 11th ACM conference on Computer and communications security

[8] Lecture Slides: Integrity Reporting / Attestation, 2009,
http://www.ei.rub.de/studierende/lehrveranstaltungen/231

[9] PrivacyCA,
http://privacyca.com/