WHITEPAPER

TPM 2.0 and Certificate-Based IoT Device Authentication

Paul Griffiths, Software Engineer, GlobalSign

In this article, we will look at how the TPM 2.0 privacy preserving protocol for distributing credentials for keys on a TPM (the "TPM Protocol") can provide the manufacturers of IoT devices and the services which use those devices with more confidence in their certificate-based device authentication processes.

Content

Introduction 2

What is a TPM? 2

Device Certification 3

The Protocol 3

Public and Private Areas 4

What the CA Does 4

What the TPM Does 5

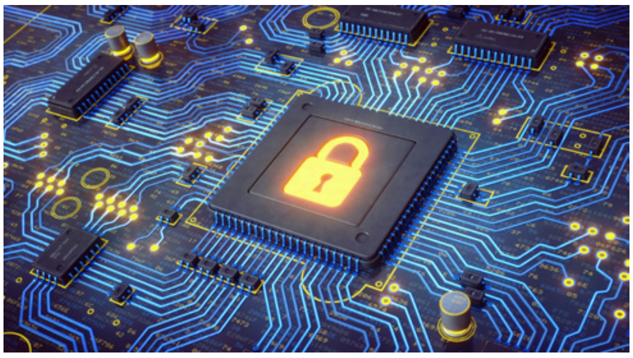
Conclusion 6

Introduction

Digital signatures, when properly implemented, can provide a strong basis for believing that a message was sent by a known sender, and that it was not altered during transmission, and these two properties are fundamental to any trusted communication. The underlying public-key cryptography can provide us with a high level of assurance that the message could only have originated with the possessor of the private key corresponding to that sender's public key. Since - in contrast with symmetric cryptography - that private key never needs to be shared with anyone else, if we are confident that the sender has appropriately protected their private key, then we can be confident that the message came from the sender, and we can use this confidence as a basis for authentication.

However, we can only have this confidence if we are sure that the public key in our possession really does belong that particular sender. If the sender is a human being who personally gave us a copy of the key then things are easy, but the estimated 30 billion devices on the Internet of things (IoT) pose a more significant problem. Many of these devices are increasingly in our homes and our vehicles, as well as in critical civilian and even military infrastructure, and their scope and number continue to grow rapidly. Compromised devices or private keys present a grave security risk, and more than ever, services for IoT devices need to be absolutely sure that they are communicating with genuine, authorized and secure devices.

The TPM Protocol can provide manufacturers of IoT devices and the services which use those devices with more confidence in their certificate-based authentication processes for IoT devices containing a TPM.



What is a TPM?

A Trusted Platform Module, or TPM, is a secure system component which, in conjunction with other system components, allows an independent entity to determine if a device's trusted computing base has been compromised. TPMs are in widespread use on modern PCs and notebooks for applications including secure boot, full disk encryption, and hardware-based password protection. The **TPM specifications** (version 2.0 being the most recent) are developed by the **Trusted Computing Group** (TCG).

For this article, the key fact is that a TPM can function as a low-cost cryptographic co-processor, providing secure, tamper-proof, hardware-based key generation and encrypted key storage. Since each TPM has a unique and secret Endorsement Key (EK), usually certified by a trusted CA, convenient authentication solutions can be developed which ensure a communicating device contains a legitimate and identifiable TPM. When combined with other TPM capabilities such as secure boot and hardware/ software attestation, the security of an IoT deployment can be greatly enhanced.

Device Certification

The EK is unique to an individual TPM. It is generated inside the TPM (or burned into the TPM by the manufacturer), and the private component is protected by the TPM hardware and cannot be read or exported from the TPM. Once we have verified the EK public component from its certificate,[1] then if we can prove an IoT device possesses the EK private component, we have positively identified the TPM.

When a device wants to authenticate itself with an online service, a common solution is to use certificate-based TLS client authentication. During the TLS handshake, the device provides a copy of its certificate to the server and digitally signs some protocol-defined information to prove that it possesses the private key corresponding to the public key in the certificate. The server verifies the certificate against its database of trusted CA certificates, and then cryptographically verifies the digital signature provided by the device. If both steps are successful, the device is authenticated.[2]

The problem is that the use of the EK is restricted by the TPM to a limited set of decryption operations. The device therefore cannot use it to create digital signatures, and so it cannot use its EK certificate for TLS client authentication in this way.

The solution is for the device to use the TPM to generate a second hardware-protected key which can be used for digital signatures, and to have that new key certified by a CA. The CA verifies the EK and, if successful, issues the device with a certificate for this new key which it can use for TLS client authentication.

This approach also has privacy benefits. Because the EK uniquely identifies the device, if the EK certificate could be used for authentication, a device's activity could be tracked and correlated across multiple services, which may be undesirable. By using secondary certificates, a separate certificate could be issued for each different service the device accesses, or even a separate certificate for each access to the same service, and privacy can be maintained.

Of course, this shifts the burden of identifying the TPM onto the CA. If the EK can't be used for digital signatures, how does the CA verify that the device possesses the private component?

The Protocol

The TCG solution is conceptually simple. Rather than asking the device to prove it possesses the EK private component before issuing a certificate, the CA essentially issues a certificate encrypted with the EK public component such that only the device possessing the EK private component will be able to decrypt it. In addition, the device will only be able to decrypt the certificate if the key being certified is also loaded on the TPM. The users of the certificate can therefore be confident not only that the device contains a valid TPM, but that the key it is using to authenticate itself is also protected by the TPM, and therefore at a lower risk of compromise.

If we are confident that the sender has appropriately protected their private key, then we can be confident that the message came from the sender

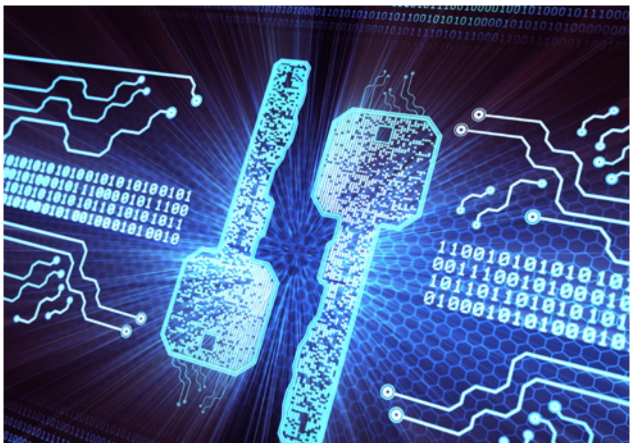
1 For the remainder of this article, we will assume we are working with a TPM with an EK certificate issued by a trusted CA.

2 The server may perform additional checks, of course.

Public and Private Areas

To understand the protocol, we first need to know that each key on a TPM has a "public area" and a "private area". For asymmetric keys, the private area contains the private key itself, and can either never be read or exported from the TPM (in the case of an EK) or can be exported only for storage in an encrypted form which can only be decrypted back on the same TPM.[3] The public area, on the other hand, does not need to be confidential[4] and is usually accessible to anyone with physical access to the TPM. It includes, among other things:

- The type of key (e.g. RSA, ECC, or a symmetric key);
- The hash algorithm used to compute its "name" (see below);
- The public component of the key, if it is an asymmetric key;
- The symmetric cryptographic algorithm it uses to encrypt child keys, if it's a storage key (the EK is always a storage key); and
- Various attributes, such as whether it can be used for signing, or decryption, or both, or whether it's capable of being duplicated to a different TPM.



The "name" of a key is actually not a name at all, but a hash of its entire public area using the specified hash algorithm. The effect of this is that if any piece of the public area changes, its "name" will also change. The significance of this will be made clear shortly.

What the CA Does

In additional to a copy of the EK certificate, the CA usually needs to receive, at a minimum, the public areas of:

- The endorsement key – because it's going to need the public key, the name hash algorithm, and the symmetric encryption algorithm;
- The key to be certified – because it needs the public key to put in the issued certificate, because it has to compute the "name", and optionally so it can examine the other attributes and decide whether or not it wants to issue a certificate for this key at all.

If it decides to issue a certificate, the CA returns a "credentials blob". This comprises two parts:

- An HMAC for authentication;
- The "credential" itself, encrypted with the symmetric algorithm specified by the EK's public area

Ideally this would be enough. However there are two problems. The first is that the "credential" is limited in size to the size of the digest produced by the endorsement key's name algorithm. With SHA256, for example, this would be 32 bytes, which is far too small to contain a certificate. Therefore the CA will encrypt the certificate separately with an entirely separate symmetric key it generates randomly, and the "credential" will actually be this symmetric key.

3 Although the private area of a key can never be read directly, it is possible to create TPM keys which can be migrated or duplicated onto another TPM. As we will see, it's possible for a CA to ensure it doesn't issue certificates to keys which can be duplicated in this way.
4 Although it can still compromise privacy if it can be linked to a specific user or device.

The second problem is that both the HMAC and the encrypted credential need a secret key to recompute/decrypt, and the TPM has neither of them. The solution is that both keys are computed from a standard key derivation function (KDF) which has various inputs. All the inputs are public (e.g. the "name" of the key to be certified, certain specific strings mandated by the TPM library specification) and already available to the TPM, except for one: a random number, or "seed", generated by the CA. To securely communicate this to the TPM, the CA will encrypt it using the EK public component. The TPM can decrypt the seed and use it with the standard KDF to compute the same HMAC and decryption keys that the CA used to create the credentials blob.

So the CA actually returns three things:

- A random seed, encrypted with the EK public component
- A credentials blob containing an HMAC and a symmetrically-encrypted credential
- A certificate, symmetrically-encrypted with this credential.

What the TPM Does

Once it's received the information from the CA, the TPM first decrypts the seed using the EK private component. It then uses this seed with the KDF to derive the same HMAC key and decryption key that the CA used to create the the credentials blob.

Note that one other input to the KDF in this case is the "name" of the key to be certified, which the TPM will obtain directly from a key it currently has loaded. The CA, of course, also used this name to derive the encryption key, and we can now see its importance. Because the name is an input to the KDF, activation of the credential will fail if the public area of the key on the TPM does not have the exact same properties that the CA was led to believe it had. For example, if the device provides the CA with a public area stating the key cannot be duplicated to a different TPM, but then tries to activate the credential for a TPM key which actually can be so duplicated, the activation will fail and the device will be unable to decrypt and retrieve the certificate. If the device requests a certificate for a key which is not protected by the TPM at all, activation will similarly fail.

Using the seed and the name of the key to be certified, therefore, the TPM:

- Uses the KDF to independently compute the secret key for the HMAC;
- Recomputes the HMAC to verify that the encrypted credential was not tampered with during transit;
- Uses the KDF to independently compute the secret key which is encrypting the credential; and
- Decrypts (or "activates") and returns the credential.

The TPM's job is now done. The device now has the credential, i.e. it has the symmetric key it needs to decrypt the actual certificate, which it can then do without the assistance of the TPM.

Note that because the EK is, in TPM terminology, a restricted decryption key, it can only be used for specific purposes defined by the TPM. In particular, a user cannot bypass the checks over the key to be certified by just manually decrypting the seed with the EK and using it to independently compute the HMAC and credential keys with the standard KDF. The TPM maintains integrity by only allowing the EK to perform the decryption in the context of the entire credential activation process.

Conclusion

The TPM Protocol enables service providers - by relying on a CA which implements it and using TPM-enabled devices - to ensure strong, hardware-based IoT device authentication, and IoT device manufacturers can give their products a competitive advantage by leveraging the capabilities of TPM 2.0.

GlobalSign's Edge Enroll service offers certificates issued under an implementation of the TPM Protocol, along with an open-source client library and application to access it. Contact our sales team for further information.

About GlobalSign

As one of the world's most deeply-rooted certificate authorities, GlobalSign is the leading provider of trusted identity and security solutions enabling businesses, large enterprises, cloud-based service providers, and IoT innovators worldwide to conduct secure online communications, manage millions of verified digital identities and automate authentication and encryption. Its high-scale PKI and identity solutions support the billions of services, devices, people, and things comprising the IoT. A subsidiary of Japan-based GMO GlobalSign Holdings K.K and GMO Internet Group, GMO GlobalSign has offices in the Americas, Europe and Asia. For more information, visit https://www.globalsign.com

GlobalSign US Office
Two International Drive
Suite 150, Portsmouth
New Hampshire 03801
Phone: 603-570-7060
Email: **sales-us@globalsign.com**

GlobalSign UK Office
Springfield House,
Sandling Road, Maidstone,
Kent ME14 2LP
Phone: 01622 766766
Email: **sales@globalsign.com**