Offline dictionary attack on TCG TPM weak authorisation data, and solution

Liqun Chen
HP Labs, UK

Mark Ryan
HP Labs, UK, and
University of Birmingham

Abstract

The Trusted Platform Module (TPM) is a hardware chip designed to enable PCs achieve greater security. Proof of possession of values known as authData is required by user processes in order to use TPM keys. We show that in certain circumstances dictionary attacks can be performed offline on authdata. In this way, an attacker can circumvent some crucial operations of the TPM, and impersonate the TPM owner to the TPM, or the TPM to its owner. For example, he can unbind data or migrate keys without possessing the required authorisation data, or fake the creation of TPM keys. This means that any application that relies on the TPM may be vulnerable to attack.

We propose a new solution and some modifications to the TPM specification to prevent the offline attacks, and we also provide the way to integrate these modifications into the TPM command architecture with minimal change. With our solution, the user can use a password-type of weak secret as their authData, and the TPM system will be still safe.

1 Introduction

The Trusted Platform Module (TPM) is a hardware chip specified by the Trusted Computing Group (TCG) industry consortium, with the aim to enable computers to achieve greater levels of security than was previously possible. There are 100 million TPMs currently in existence, mostly in high-end laptops. The TPM stores cryptographic keys and other sensitive data in its shielded non-volatile memory. Application software such as Microsoft's BitLocker and HP's HP ProtectTools use the TPM in order to guarantee security properties.

The Trusted Platform Module (TPM) stores cryptographic keys and other sensitive information in shielded locations. Processes running on the host laptop or on other computers can use those keys in controlled ways. To do so, such processes have to prove knowledge of the relevant authorisation data, called authData. The authData is chosen by the user process, and sent encrypted to the TPM. The TPM stores the authData along with the relevant keys or other sensitive information. In any communication between the user and TPM which requires owner authorisation, the authData is used as a HMAC key.

Although authData is 160 bits and is therefore capable of being a high-entropy value, it may be that actual authData is derived from human-memorable passwords, and is therefore low-entropy. The TPM specification [5] has no restrictions about the entropy expected in authData. If low-entropy data is used, an attacker could try successively to guess all the possible values of the authData, and verify each guess in turn. Such attacks are called dictionary

attacks. The TPM specification stipulates that TPM manufacturers should implement resistance to dictionary attacks on authorisation data, for example, by permitting only a small number of incorrect guesses per minute.

We show that guesses of low-entropy data can be verified offline, so that the resistance offered by the TPM is ineffective. We propose a new solution and modifications to the TPM specification to prevent the offline dictionary attacks, and we also provide the way to integrate these modifications into the TPM command architecture with minimal change. With our solution, the user can use a password-type weak secret as their authData, and the TPM system will be still safe.

2 The offline dictionary attack

We show that an attacker that can observe some dataflow between the TPM and the user processes can perform an offline dictionary attack. Specifically, if the attacker can observe:

- A command containing proof of possession of authData (typically during an *Object-Independent Authorization Protocol* (OIAP) session), and the TPM response;
- Or, a command containing proof of possession of the shared secret associated with such authData (typically during an *Object-Specific Authorization Protocol* (OSAP) session), and the TPM response;

then the attacker can conduct an offline dictionary attack to discover the authData. This is possible because the attacker can confirm its guess of authData by reconstructing the authorisation HMACs based on the guessed authData, and comparing with the observed HMAC. If they are equal, that confirms the guess. The TCG specification assumes that the required traffic observations by the attacker are possible, and was intended to protect against them.

Moreover, the TPM specification uses shared secrets derived from authData to protect new authData supplied by the user for newly created keys. Once a single piece of authData is compromised, any new authData protected by it is also compromised. Therefore the technology specified in the current TPM specification version 1.2 can only work safely with the condition that users always choose a strong secret as their authData and never disclose their authData to any mistrusted entities. Obviously this condition restricts the TPM applications.

We propose a new solution and modifications to the TPM specification to prevent the offline dictionary attacks, and we also provide the way to integrate these modifications into the TPM command architecture with minimal change. With our solution, the user can use a password-type weak secret as their authData, and the TPM system will be still safe.

3 Password-based key agreement

Password-based authenticated key agreement is a cryptographic primitive that addresses this kind of problem. It is specified in IEEE P1363.2 [1] and ISO/IEC 11770-4 [2]. There are a large number of such key agreement protocols. The most attractive one for the current purpose is SPEKE (Simple Password Exponential Key Exchange) by Jablon [3, 4]. It enables a pair of entities A and B to establish a strong shared secret s based on a weak secret w that they already share.

To do this, they run a Diffie-Hellman key exchange protocol, in which they use the shared weak password to compute a group generator. Let G be a finite field group of prime order q and prime modulus p, where q is at least 160 bits, and p is at least 1024 bits, such that $q \mid p-1$. Let H be a secure hash function $H : \{0,1\}^* \rightarrow G$. We assume that the values q, p and the function H are known to A and B (they are not secrets), and also that the weak shared secret w is known to A and B. The exchange proceeds as follows.

- A creates a new random $x \in Z_q^*$ and sends $H(w)^x \bmod p$ to B. For simplicity, we omit "mod p" in $H(a)^b \bmod p$ for any values a and b in the remaining part of the paper.
- A responds by creating a new random $y \in Z_q^*$ and sends $H(w)^y$ to B.
- Now, A and B can each compute the strong shared secret as $s = H(w)^{xy}$.

4 Solving the offline authData attack

Our proposed new method of TPM resistance to offline dictionary attack on weak authorisation data has been developed based on the SPEKE protocol. We assume that the user process can introduce new authData to the TPM in a reliable way (e.g., by encrypting it with an already established TPM key). It remains to demonstrate how the user process can later prove its knowledge of the authData when it wants to execute an authorised command.

In our basic solution, we directly make use of the SPEKE scheme between the TPM and user process to derive a strong secret based on the weak authData. SPEKE applied to this situation would look as follows. Every time a user process executes a command requiring authorisation with authData d, the user process and the TPM engage in the SPEKE protocol using d as the weak shared secret w. The user process chooses a random x and sends $H(d)^x$ to the TPM; the TPM chooses a random y and sends $H(d)^y$ to the user. Then they each compute the strong secret $s = H(d)^{xy}$ as explained above. After that, the value w is replaced by the value s as a HMAC key.

We have a number alternatives of the above basic solution, each of which achieve a unique requirement, which might be needed by different applications. We will give the details in the full paper.

Alternative 1: Password-based key retrieval. The aim of this version is to reduce the TPM's computation task in the basic solution. Instead of choosing a fresh y each time, the TPM has a long-term secret key y, called the "authData key", which is used to process multiple authData values. At the time the user process sends the encrypted newly chosen authData d, the TPM stores d and returns the value $H(d,t)^y$ where t is some object-specific text (such as the name or the digest of public key of the object). Every time a user process executes a command requiring authorisation, it creates a new random x and sends $H(d,t)^x$ to the TPM, together with the command requiring authorisation using $H(d,t)^{xy}$ as the HMAC key.

The object-specific text t is required to avoid an *online* attack. Without t, an attacker could use the TPM as an oracle to confirm his guess d' of some authData d by introducing a new object with authData d', and comparing $H(d)^y$ with $H(d')^y$. The TPM should not resist such an attack.

Alternative 2. Password-based proof of knowledge. The aim of this solution is to avoid the TPM's long-term authData key being directly used by multiple users in multiple sessions, in order to enhance safety of the key. Again, the TPM has a long-term secret key y. At the time the user process sends the encrypted newly chosen authData d, the TPM stores the value $k = H_0(d, y, t)$ where H_0 is a secure hash-function $H_0 : \{0,1\}^* \rightarrow Z_q$ and t is the object-specific text; then the TPM replies with the message $H(d)^k$, and discards d. To demonstrate authorisation for a command, the user process chooses a random x and sends $H(d)^x$, and uses the value $H(d)^{kx}$ as the HMAC key for the command requiring authorisation. The user process may use the same x across several authorisations, or it may pick a new x each time.

Alternative 3. Password-based proof of knowledge without long term key. The aim of this solution is to further reduce the TPM's computation and storage task in the previous solutions. At the time the user process sends the encrypted newly chosen authData d, the TPM chooses a random value $k \in Z_q^*$ and stores it. The TPM replies with the message $H(d)^k \in G$, and discards d. To demonstrate authorisation for a command, the user process chooses a random x and sends $H(d)^x \in G$, and uses the value $H(d)^{kx} \in G$ as the HMAC key for the command requiring authorisation. The user process may use the same x across several authorisations, or it may pick a new x each time.

5 Integration with TPM command architecture

We show how to integrate our solutions into the TPM command architecture, requiring minimal changes to the existing command set. We illustrate that for "Alternative 3". The changes listed are described from the point of view of the TPM:

1. Commands that introduce newly created authData require to be changed. The incoming and outgoing operands and their sizes do not need to be changed, but the TPM should not store the authData d. In the place of d it stores the new random k that it created, and it discards d.

2. Commands that require proof of possession of authData also require to be changed. The value $H(d)^x$ computed by the user process should be supplied as an additional incoming operand. The TPM then retrieves the value k that it stored in place of the authData, and it uses $H(d)^{kx}$ in the HMAC key in order to reconstruct and verify the incoming HMAC.

6 Conclusion

We have proposed a new solution for TCG TPM resistance to offline dictionary attack on weak authrisation data. The new solution offers the following advantages over the existing solutions: (1) The new solution protects the weak authorisation data from offline dictionary attacks. (2) The new solution can be integrated into the TPM command architecture, requiring minimal changes to the existing command set.

Acknowledgments. Many thanks to Carsten Rudolph for pointing out the necessity of the object-specific text t in alternatives 1 and 2.

References

[1] IEEE P1363.2/D26 Draft Standard for Specifications for Password-based Public Key Cryptographic Techniques. `grouper.ieee.org/groups/1363/passwdPK/index.html`

[2] ISO/IEC 11770-4:2006. Information technology – Security techniques – Key management – Part 4: Mechanisms based on weak secrets.

[3] David Jablon. Strong password-only authenticated key exchange. *Computer Communication Review* **26**(5):5–26. ACM SIGCOMM. October 1996.

[4] David Jablon. Extended password key exchange protocols immune to dictionary attack. In *Proceedings of the Sixth Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET-ICE '97)*, pages 248–255. IEEE Computer Society, 1997.

[5] Trusted Computing Group. `www.trustedcomputinggroup.org`