SHA-1 Uses in TPM v1.2

April 2, 2010
Ken Goldman (IBM Research), Stan Potter (United States Department of Defense)

Executive Summary

This document catalogs the usages of SHA-1 in the TPM v1.2 specifications and assesses the impact of the collision attacks discovered in 2005 against them. In general the TCG took several precautions to guard against collision attacks, such as concatenating fresh randomly generated values or randomly generated values unknown to the outside to the data before hashing. Also, the use of fixed structures in the data impedes the ability of an attacker to manipulate the data to effectively carry out a collision attack.

We did find one corner case in which the RSA signature command may hash the input before signing. In this case the TCG inadvertently chose a value outside of the TPM's control to concatenate to the data before signing. Had they been consistent in choosing a fresh randomly generated value, there would be no problem here. Otherwise, the TPM uses SHA-1 in a manner that protects the TPM and its responses from the SHA-1 collision attacks.

General purpose SHA Engine

The TPM contains the following commands to implement a general purpose SHA-1 engine:

- TPM_SHA1Start
- TPM_SHA1Update
- TPM_SHA1Complete
- TPM_SHA1CompleteExtend

This capability facilitates the process of calculating a SHA-1 hash. The TPM exposes these commands as a convenience to platforms in a mode with insufficient memory to perform SHA-1 themselves. The TPM disallows the processing of any command during the execution of a SHA-1 session. Between the execution of TPM_SHA1Start and either TPM_SHA1Complete or TPM_SHA1CompleteExtend, the TPM will invalidate the SHA-1 session upon receipt of any command other than TPM_SHA1CompleteExtend.

Impact: Since this is a service, namely a SHA-1 engine, the impact of the collision attack on the engine itself is indeterminate. The SHA-1 engine is a necessary component of the TPM. The following sections describe how the TPM uses this service. We will discuss the impact of the collision attack for each these scenarios.

PCR Extend

The TPM_Extend command use SHA-1 in conjunction with the Platform Configuration Registers (PCRs). The input is 20 bytes, normally a SHA-1 digest computed outside the TPM. The TPM uses SHA-1 to extend one PCR in the following way:

$$New\_PCR\_Value = \text{SHA-1}(Old\_PCR\_Value \,||\, Input\_Digest).$$

Impact: The input parameters to the TPM_Extend are highly structured. Since the attacker is limited to 20 bytes, less than a full block, he/she cannot manipulate the input of SHA-1 to successfully carry out the described collision attacks.

Audit digests

With the exception of the audit commands, the TPM can audit any command. The TPM administrator (owner) can control which commands it audits. When the TPM command executes an auditable command (28 of 123 commands are auditable by default), it uses SHA-1 to hash the input parameters of the command, then extends the audit digest register, which was set to NULL either during TPM_Startup or when the previous audit session was closed, in the same manner as PCRs are extended (see previous section). The digest also incorporates a tag and TPM_COUNTER_VALUE structure, which adds more structure. When the TPM responds to an auditable command, it also hashes the output parameters, and extends the audit digest register in the same manner as PCRs are extended. All hashing and extending operations use SHA-1. The five audit commands do not use the SHA-1 engine.

Impact: All TPM commands and responses are highly structured. Even if an attacker had the ability to alter the incoming parameters of a command, he/she cannot manipulate the response of the TPM command. An attacker cannot control the output parameters of the TPM responses, and therefore cannot successfully carry out the collision attacks as described.

PCR Quotes

The TPM has two PCR Quote commands, TPM_Quote and TPM_Quote2, which use SHA-1 in conjunction with the Platform Configuration Registers (PCRs). The input is an index of at most 5 bytes, which selects a subset of the PCRs. The TPM fills a structure with the selected PCRs, hashes the structure with SHA-1 and signs the digest. The output is the signed SHA-1 digest of the selected PCRs

Impact: The input parameters to the PCR Quote commands are highly structured. A potential attacker is limited to manipulating 5 bytes, which in turn selects a subset of PCRs. Even if the attacker can successfully manipulate the PCRs through the use of the PCR Extend command, TPM_Extend, such action will trigger other security features that rely on these PCRs. Manipulation of the PCRs through the TPM_Extend command is equivalent to a preimage attack, which has not been solved for SHA-1. Although the quote structure may contain a 20-byte field of external data that the attacker can use, it is too small to mount a practical attack. An attacker cannot manipulate the input of SHA-1 in the PCR Quote commands to successfully carry out the described collision attacks.

RSA signatures

The TPM has one RSA signature command, TPM_Sign, which may use SHA-1 before applying a signature algorithm. This command handles three different cases. In the first two cases, the size of the data presented is less than the size of the signing key, and thus the TPM does not hash the data. In the last case, the TPM creates a structure, which consists of the string “SIGN”, an anti-replay nonce, and the data to be signed. It SHA-1 hashes this structure and signs the digest which it outputs in the response.

Impact: Generally speaking the inclusion of the anti-replay nonce should have thwarted potential attacks. Unfortunately, the TCG chose a nonce under the control of the attacker instead of a fresh randomly generated nonce from the TPM. In this case, an attacker could potentially manipulate the input to this function and create signatures susceptible to collision attacks. However, the integrity of the TPM and the security of the remaining functions remain intact.

Cryptographic Key Structures (Blobs)

The following cryptographic key structures contain digests calculated with SHA-1.

- TPM_STORE_ASYMKEY
- TPM_SEALED_DATA
- TPM_MIGRATE_ASYMKEY
- TPM_CERTIFY_INFO
- TPM_CERTIFY_INFO2
- TPM_DELEGATE_KEY_BLOB

These key structures frequently imbed digests of public areas inside encrypted areas, as well as digests of entire key structures, providing statistically unique identifiers.

Impact: Some of the digests protect fixed public structures which severely limits an attacker's ability to manipulate. Some of the digests are over encrypted areas in which the ability to manipulate is equivalent to breaking RSA. Some of the digests are embedded in the encrypted area. An attacker cannot successfully manipulate the input to SHA-1 for the digests in these structures to successfully mount a collision attack.

IV and Counter Values for Symmetric Algorithms

The TPM creates Initialization Vectors (IV) and initial counter values for modes of symmetric encryption that requires them using SHA-1 over a randomly generated value from the TPM and a user-supplied value.

Impact: Since the input to SHA-1 includes a random value generated by the TPM, an attacker cannot successfully manipulate the input to successfully mount a collision attack and thus control the IV or the counter values for symmetric encryption algorithms.

Authorization Sessions

The TPM establishes and maintains authorization sessions. It uses SHA-1 to provide integrity and anti-replay features through HMAC.

Impact: Since the TPM uses fresh randomly generated nonces from the TPM in each HMAC calculation, these sessions are protected against SHA-1 collision attacks.

DAA

The TPM uses SHA-1 to verify digests of the DAA algorithm itself, as well as fixed structures within the protocol. It also uses SHA-1 to hash algorithm parameters and fixed structures.

Impact: DAA parameters are a fixed size as well as nonrandom values. The structures are also fixed. An attack has no opportunity for manipulating the parameters and

structures to successfully mount a collision attack without causing the DAA protocol to completely collapse.

Identity Keys

TPMs can create and activate identity keys. While activating an identity key the TPM creates and validates digests of fixed key structures and other fixed structures.

Impact: Since the structures are fixed, an attacker has no opportunity for manipulating them for a successful collision attack.

Transport Sessions

The TPM's transport sessions have a logging option. Even without the logging option, the commands for transport sessions have numerous calls to create digests with SHA-1.

Impact: In all cases, the inputs to SHA-1 are fixed structures. An attacker has no opportunity to mount a successful collision attack during transport sessions.

Saving TPM Resources through Context Commands

A TPM user can save off TPM resources by using the TPM_SAVECONTEXT command and reload the resources using the TPM_LOADCONTEXT command. The TPM uses SHA-1 to compute the integrity digest over the entire context before encrypting the sensitive area and includes a randomly generated value unknown to the attacker. It validates the digest as it reloads the resources.

Impact: Since the TPM computes a digest before encrypting the sensitive area and uses a random value unknown to the attacker, these techniques effectively block collision attacks.

Delegation

The TPM allows the delegation of individual commands that require administrative (Owner) privilege to other users without giving them full access to all privileged commands. One may also delegate keys as well through key authorization. It uses SHA-1 to calculate digests of the delegation tables, to verify these digests, to create a key stream, and to create IVs for symmetric encryption.

Impact: Since the key stream is used to protect authorization values, which is not required, we will not consider this case. The IVs are created using fresh randomly generated nonce, which protects it against collision attacks. Finally, the delegation blobs are highly structured and their hashes include a randomly generated number unknown to the end user, these digests are safe against collision attacks.

Migration

The TPM allows migration of certain keys to other locations. There's also a complicated use within "certified migration" of keys from one TPM through another, sometimes directly and sometimes through a third party. It uses SHA-1 to compute the hash of structures and to validate digests.

Impact: Since the digest of the migration authorization includes a randomly generated value not available to an attacker, this is protected from collision attacks. The digests of keys and key structures are highly structured, and the private areas are encrypted. Manipulating an encrypted area to successfully mount a collision attack is equivalent to breaking 2048-bit RSA, which has not been solved. Therefore an attacker would be unable to successfully carry out a collision attack using key digests. By extension, the same holds true for digests of composites of key digests.

References

Ken Goldman, e-mail dated July 7, 2006

Michael Szydlo, "SHA1 Collisions can be Found in 2^63 Operations", http://www.rsa.com/rsalabs/node.asp?id=2927, RSA Laboratories, August 19, 2005.

TCG, "TPM Main Part 1 Design Principles", v1.2, revision 103, October 26, 2006

TCG, "TPM Main Part 2 TPM Structures", v1.2, revision 103, October 26, 2006

TCG, "TPM Main Part 3 Commands", v1.2, revision 103, October 26, 2006

Wikipedia, "SHA Hash Functions", http://en.wikipedia.org/wiki/SHA_hash_functions, March 31, 2010