important to design a hardware module that increased the cost of a PC by $1 or less to allow for widespread adoption. Thus, the TPM was designed as a passive chip (it responds only to commands from platform software) that helps platform software to verify itself, as well as a

2.0 ADBI ED C IG

Although the TPM 1.1b and 1.2 specifications created the foundation for the use cases described, a number of roadblocks occurred that hampered adoption by industry. These included ambiguities in the specification, rigidness in the policy regarding how keys are used, tight coupling to specific cryptographic algorithms, and issues related to control of the TPM by platform software.

A A

Ideally, a flexible cryptographic module would handle both symmetric and asymmetric encryption algorithms. Symmetric encryption refers to algorithms where the same key is used for encryption and decryption. Asymmetric encryption (or public key cryptography1o

various physical TPMs. This will ensure interoperability among the various implementations. These are likely to include 1024-bit and 2048-bit RSA, 256-bit elliptic curve cryptography, SHA-1, SHA-256, and 128-bit AES in the first implementations. RSA and SHA-1 will be used for backward compatibility with [illegible] to in

)- 15.924322523ActualText<FEFF00A0>>> ((-10.9/T1_005)1)1051 0>Bd-0.02(•.•)-61213 7()- 0030

E A

In TPM 1.2, authorization (the process by which software proves to the TPM that it is allowed to use a key, counter, or NVRAM object) is very limited in scope. The only way to restrict access is by passwords (represented as SHA-1 hashes) and PCR values. To use a key, software will have to prove knowledge of the password hash as part of the command. It is also possible to seal a key to specific PCR values so that the key cannot be used unless the password is known and the PCRs are in the chosen state.

This means that TPM 1.2 authorization is fairly inflexible. When multiple users share a platform, it makes it difficult to share TPM keys and data. Users often have their own sets of keys, because they individually know their passwords. This also makes system administration difficult because users must physically enter their passwords to authorize usage of their keys.

Enhanced Authorization (EA) in TPM 2.0 greatly expands the methods by which key and data use can be authorized, and the policy has become much more flexible. In TPM 1.2, software would prove it had knowledge of the password in an authorization session. A single command would be sent to the TPM before the command requiring authorization to start the session. One of the parameters to the command was a hashed message authentication code that included the password hash along with other values so that the TPM could verify knowledge of the password.

EA extends these authorization sessions into policy sessions, which allow multiple authorization methods to be combined by Boolean logic. For instance, let's say one wants to have a key that is accessible by both Alice and Bob and they have individual passwords. A policy can be created that says, "Authorize access if and only if Password(Alice) or Password(Bob)." Or say one wanted to create multifactor authentication for both users but allow either user to access the key. A policy can be created that says, "Authorize access if and only if ((Password(Alice) and SmartCard(Alice)) or (Password(Bob) and SmartCard(Bob))." The way this works is that software will create the policy and then specify the hash of that policy when creating the key or data. The TPM does not need to know the details of the policy—the hash is sufficient. Later, when software wants to use the key, it will start an authorization session and send one command to the TPM for each token in the policy equation. The TPM will verify that the policy specified in that sequence of commands is satisfied and also verify that the hash of the policy command sequence is the same as the policy hash specified at creation time.

Not only does EA create a flexible policy language, it also adds more possible authorization methods. Previously, one was limited to passwords and PCR values. Now the authorization methods include the following:

- Passwords—similar to TPM 1.2
- PCR values—same as TPM 1.2, but the addition of Boolean logic means multiple PCR states can be used
- TPM counter or NVRAM value—require that these items have a particular value
- Physical presence— require that a user be physically present at the PC
- Commands—require that the object can only be used with a given set of commands
- Digital signature from a public key—this allows smartcards such as Common Access Cards to be used for authorization

Management for some things is much easier to do with simple passwords than with complex policies, and so the design allows for each object to have a simple password associated with it, a policy, or both. If both are present, the creator of the object can split roles between the two, so that (for example) a simple password may be used for signing with a key, but administrative tasks such as creating a certificate for the key or creating a backup of the key may require an IT administrator's smartcard authorization.

C

In the TPM

14