To:	INCITS T10 Committee
From:	David Black, EMC
Date:	6 November 2006
Document:	T10/06-493r0
Subject:	SSC-3: Review of T10/06-389r1: Using Public-Key Cryptography for Key Wrapping

I arranged for a review of T10/06-389r1 by some RSA Security colleagues. Their review comments are in Courier, my additional notes are in blue Times New Roman, tagged with my name.

```
We're not familiar with ECIES-KEM per ISO 18032-2; RSAES-OAEP (the
alternative approach cited) remains an acceptable method, but we've
recently preferred (RSA) KEM per X9.44.
```

David Black: This preference is for what would be used in a new design by RSA Security. Further clarification: "Either RSAES-OAEP or RSA-KEM are suitable choices; there are possible theoretical advantages for RSA-KEM, hence our current preference, but either is acceptable."

```
- Caveat: these thoughts don't attempt or represent a qualified,
comprehensive assessment of the suitability of the approach to satisfy
application and environmental requirements and constraints for the
problem domain.
```

David Black: In particular, the RSA Security reviewers are not SCSI experts.

```
- Per 4.1.1 and 4.1.2, the general approach wraps a key using a target
device's public key, and optionally authenticates that wrapped key by
signing the wrapped result using a signer's key.  As described, it's
also possible (though involving means apparently outside the scope of
this contribution) to obtain a nonce from the device and use it within
the signature operation to ensure freshness.  Is it ever valid for the
wrapping and signing devices to be different (i.e., for the master
backup server to wrap and the backup server (which would receive any
nonce) to sign)? If not, a receiver should explicitly confirm that they
match. If so (as 8.5.4.8 suggests), the device will need to know which
signing entities correspond to a given wrapping entity.
```

David Black: Current language in the proposal appears to require the wrapper and signer to be the same entity. Separating them probably involves additional support for communicating identity information (to support the "receiver should explicitly confirm" statement above). The identity support in the current form of this proposal is minimal.

```
- per 4.2, it's unclear from this contribution alone whether the
requirement for a wrapping entity to support all formats implies a
universal need to support ECC crypto capabilities.  I don't think this
should be functionally necessary, and it could be valuable to avoid a
uniform ECC support requirement to avoid unnecessary IP encumbrances.
```

David Black: Certicom is an important holder of ECC (Elliptic Curve Cryptography) patents. See: http://www.certicom.com/index.php?action=ip,overview

- per Sec. 4.3, it's unusual to describe installation of a device's public key in a key-wrapping entity as a means that confers authorization to send to that device. Public keys are normally assumed to be public, and protection methods (e.g., hardware modules) are oriented to protecting the confidentiality of private keys. An approach relying on a device's ability to authenticate the wrapping entity, and to compare that authenticated identity with an authorized set, would be more conventional.

David Black: This is polite. The proposed authorization approach requires that a device server's public key be treated as a secret. That's a bad idea.

David Black: I also have a concern that the proposal uses public keys (actually fingerprints of public keys) as identities - that creates security management and infrastructure integration issues. Addressing this concern probably involves the introduction of separate identities and support for certificates. One example is that use of the proposed ECIES-KEM mechanism in a larger system where something other than public key fingerprints are used for identities is a probable violation of at least the intent (and possibly the letter) of the NIST SP800-56A KDF specification.

David Black: The upshot is that an entity should assert an identity, prove knowledge of the secret corresponding to that identity (authentication) and then have its privilege to perform the requested operation checked (authorization).

- per 4.4.1, item 1 in the assumptions list towards the end of the section, SSL ciphersuites beyond common current practice would be needed to achieve a security level consistent with some of the tape-level crypto they're contemplating.

David Black: Specifically, there are no currently defined SSL or TLS ciphersuites with strength equivalent to a 256 bit AES key.

- per 4.4.1, item 4 in the assumptions list, can it be assumed that the tape drive's module can retain multiple server keys, as to distinguish backup server/master backup server/key manager identities and/or to accomplish graceful key rollover?

David Black: This needs to be stated explicitly, with a minimum requirement for how many keys a device server shall be able to retain simultaneously. FC-SP had a "lengthy discussion" about exactly this issue.

- per 4.4.2, "Key Disclosure (passive)", "Against Backup Server" column: is the intent here to encrypt keys for transfer via the wrapping discussed in this document or via some separate means?

David Black: At a minimum, this indicates a need to coordinate this with the key management work underway in TCG; TCG ought to be the primary venue for specifying key wrap formats for use by a key management server.

- per 4.4.2, last row in Table 1, signing key rotation doesn't seem like a direct and effective countermeasure against this threat, as signing key cryptoperiods would probably be rather long in the absence of extensive key management to frequently replace keys representing signers. The nonce approach suggested in 4.1.2 may be useful here.

David Black: Again, this is polite - the blunt version is that key rotation is not an effective anti-replay countermeasure.

- per 4.4.2, penult. para.: I'm not wholly clear about which level of encryption within the system might be turned off, or what attacks they're envisioning here as enabling this. Clearly, though, this is a critical concern to be able to defend against.

- 4.4.3 proposes encryption and signing at the key server. If the nonce-based strategy proposed in 4.1.2 is adopted, this seems to imply that all key transactions with backup servers must reach up the hierarchy via protocol hops to that key server. Is this implication acceptable?

David Black: In other words, this would require the key server to be online and involved in the transmission of a key to the device (similar to Kerberos - the server must be up and available). One of the advantages of public key cryptography with PKI is that the Certification Authority server does not need to be contacted in order to use a certificate; this is not an advantage to be discarded lightly. One possible alternative is to sign the wrapped key twice, once at the key server ("<X> wrapped this key") and once at the backup server ("<Y> is putting this key into the device").

- per sec. 5.1.2, ECC-521 seems quite large relative to the alternative of RSA-2048. (NIST 800-57, e.g., considers 512-bit ECC as comparable to RSA-15360, at the overall 256-bit symmetric strength level; given that 128-bit strength is considered there as suitable beyond the year 2030, the 256-bit level seems like computationally expensive overkill for most requirements.) More broadly, suggest that a range of key lengths should be supported in each family and that agility be supported for hash and encryption algorithms generally.

David Black: This is a version of an issue encountered by 06-103. There are a lot of reasonable algorithms possible in this space, and restricting to exactly 2 choices is unlikely to satisfy all (or even most) involved.