Comments received on SP 800-57
Recommendation for Key Management, Part 1: General Guideline

NOTES ON NIST SPECIAL PUB 800-57

(Nechvatal, 2/12/03)

1. p. 1: "page 99 ... 112 bit" should be "112 bits".

2. p. 3: "Arithmetic" in entry 4.2.5.1 should be handled as in 4.2.5.2.

3. p. 7: entries B.3.14.10,11: extra spaces should be eliminated.

4. Sec 1.4, last paragraph: "strong cryptography" should be defined, or an example of weak cryptography provided.

5. Glossary, p. 18, "key share ... will not be addressed" and p. 21, "split knowledge," which is addressed. This seems inconsistent.

6. Glossary, p. 19, "operational **": these two entries both refer to "lifecycle" of keying material; it should be explained how "lifecycle" differs from "cryptoperiod."

7. Glossary, p. 22, "statistically unique:" the definition here should be consistent with the corresponding definition on p. 9 of Spec. Pub.
800-56.

8. Glossary, p. 22, "Work:" last sentence: "computer" should be "computers."

9. Sec. 4.2.4, last sentence (brackets): "is planned" should probably be something like "is planned; see the next section."

10. Sec. 4.2.4.1, first sentence: for consistency with Spec. Pub. 800-56, it would be nice to add FIPS 186-3 to the references.

11. Sec. 4.2.4.2, second sentence; Sec. 5.1.1, p. 37, top; Sec. 8.3, p. 87, third paragraph; Sec. 8.8.1, #5; and Sec. 8.8.2, #5: "[FIPS 186-3]" looks like a dangling reference (not listed in references).

12. p. 31: footnotes 3 and 4 are identical. This looks superfluous (can a reference to a footnote be repeated?).

13. p. 31, footnote 5: does it serve any purpose to introduce the symbols f and n here? They are defined again in Sec. 8.8.1, p. 96, whence p. 31 will be long forgotten.

14. Sec. 7, p. 52, first sentence: uses "life cycle" of keys, as opposed to "lifecycle" of keying material (p. 19) and keys (p. 75), "lifetime" (p. 58) of secret values, "lifespan" of data (p. 78), and "security life" of data (p. 101).

15. Sec. 7.3.4 and footnote 11: this is a very dangerous combination. The text in Sec. 7.3.4 is fine but vague: it puts the onus on the user to erase keying material so that it "cannot be recovered," which is pretty definite. However, footnote 11 seems to imply that one overwrite of a key (e.g., on a hard disc) will make it unrecoverable, which is false. More importantly, the only user-controlled operation on a disc is "rm" or its equivalent, which of course doesn't actually erase it. I have absolutely no idea how I would get rid of a key so that it is unrecoverable. I doubt if the situation is much different for a module such as a smart card. The same issue arises on pp. 76 (Sec. 8.1, #3) and 79 (#3): the user is told that he SHALL destroy the key, but is not told how to really accomplish this.

16. Sec. 8.8.1, third paragraph: "equivalencies:" is that a word?

17. Sec. 8.8.2, third paragraph: I'm curious about who determined that 80 bits of security is good enough til 2015. Factoring 700-bit numbers or breaking DSA at p of 700 bits involves a workfactor in the ballpark of 2**80, so I wouldn't trust 80 bits of security even today (see my paper "Security of RSA and the DSA," 5/5/94).

18. Sec. 8.8.2, p. 99, below Table 9, bracketed question: "for 112 bits, should SHA-256 be truncated?" The answer should be consistent with FIPS 800-56, p. 26, Sec. 5.6, Table 3, where "SHA224" is defined.

19. Appen. C, 4th paragraph: "Using a key generated a key" ??

From: "Kaliski, Burt" <BKaliski@rsasecurity.com>
To: "'GuidelineComments@nist.gov'" <GuidelineComments@nist.gov>
Cc: "Linn, John" <jlinn@rsasecurity.com>
Subject: RSA Security Comments on NIST SP 800-57
Date: Fri, 28 Mar 2003 16:51:13 -0500

RSA Security Comments on NIST Special Publication 800-57, "Recommendation for Key Management - Part 1: General Guideline", January 2003 draft

Burt Kaliski and John Linn
{bkaliski, jlinn}@rsasecurity.com

March 28, 2003

General Comments

This document provides a valuable service by structuring and presenting many concepts relevant to key management in a structured, systematic context. It will provide a valuable reference for implementers of systems based on cryptography.

We endorse the document's apparent goal of accommodating accepted commercial practice in many areas, complemented with FIPS-140-2 assurance for sensitivecryptographic processing.

Specific Comments

Sec. 2.1, p. 14, discussions of Access Control and Authorization should note that privileges may be granted, e.g., to roles or transferred as capabilities, rather than being bound to specific identified entities.

p. 15, definition of Cryptographic Module implies need to include definition of Cryptographic Boundary.

p. 17, definitions: should include a definition for Key.

p. 17, Key de-registration: is this intended to prohibit retention of an audit trail?

p. 18, Key update: this definition seems to conflict with the more conventional usage in Sec. 7.2.3.2. Rather than being a lifecycle stage, key update seems more like a transform operation, which may in some cases be performed multiple times.

p. 19, Private key, 2nd sentence: per definition of "Owner" above, the trusted key generating party may also know the private key. Item 1: While possible for some algorithms, use of a private key to compute a public key is unusual.

p. 22, Work: suggest adding, at end, ", assuming that the algorithm being executed can be sufficiently parallelized." In the same line, "such computer" -> "such computers".

p. 24, 3.1, 2nd sentence. While there is precedent for use of the term "privacy" as a synonym for confidentiality, this usage appears to be becoming less common than in the past. More commonly, "privacy" as commonly used today concerns means to constrain use or access to a user's personal data, and/or to impede correlation among a user's various actions and linkage between those actions and a human individual. While some of these concerns are addressed in part by cryptographic confidentiality, it may be worth noting the divergence.

p. 24, 3.1, general: In some applications, a confidentiality service may be compromised unless it is combined with appropriate integrity protection. As an example, Vaudenay has given an attack on the decryption operation of CBC mode with a certain padding format, whereby the attacker learns information about the plaintext from error messages returned by the decryption operation. Although this mode can be considered a confidentiality service, in practice it is not effective without an integrity protection service. This concern might be noted here.

p. 25, Authorization: This discussion should admit the prospect, e.g., of authorization based on capabilities, where authentication of an identity is not a prerequisite to obtain access rights.

p. 29, 4.2.1, 2nd para, last sentence: "well-designed hash function" -> "well-designed cryptographic hash function"?

p. 29, 4.2.1: Suggest to mention the numbering scheme, i.e., SHA-1 ends in "-1" because it was the first revision of a previous algorithm, SHA; but it could be called "SHA-160" if the previous name were not already established.

p. 30, 2nd line: "with some sort of feedback" -> "with variable IVs and some sort of feedback"?

p. 34, 5.1, 1st para: discussion should admit the prospect of keeping current and valid keys off-line in usage contexts where they are not used continuously. This can provide useful security enhancement, e.g., for CAs that need to apply signatures only on relatively infrequent occasions.

p. 34, 5.1, 4th para: It is not clear how one protects certain associations. For instance, the association between a public signature verification key and the signed data is not typically protected. The signed data may (and should) refer to the public signature verification key, but there is typically no assurance that the _correct_ key is referenced. As noted elsewhere in these comments, there may be more than one public key that correctly verifies a given signature, so signature verification does not provide assurance that the public key is the correct one. A similar concern applies to other associations with operational data:

* authenticated data for symmetric authentication keys
* encrypted data for symmetric data encryption and key wrapping keys

Perhaps some key identifier is intended?

p. 36, item d at bottom: text seems garbled (what shall be protected?).

p. 37, item e: In some sense, validity assurance is needed for nearly all keys, whether public, private or secret. Thus, one would expect the "validity assurance" column to be almost fully populated. While secret keys and private keys are easy to check for validity, the user of the key still needs the assurance that the check has been performed or that the keys are otherwise known to be valid. If the secret key or private key is generated by a third party and stored in a security module, for instance, the user cannot check the key directly.

Some comments on this column of the table:

* "For association with [private key]" presumably refers to assurance that the public key corresponds to the user's private key, as evidenced in proof of possession (POP). This provides partial assurance of public-key validity, namely that the public key has a corresponding private key (for a given signature operation). Perhaps "of association with ..." would be better?

* Validity assurance is "Yes" for the three public keys involved in key establishment, and "partial" (as just noted) for the two public keys involved in signature / authentication. We would expect authorization public keys to be treated similarly to signature / authentication private keys, yet the validity assurance

* The table's varying treatment of public keys of different types contrasts with the approach in 7.1.5.1.1 where all five types are treated the same (see items a-e on p. 56)

p. 35, items 10,11: Would there be any situation where one might want an ephemeral key transport key? Also, as to terminology: if a key transport scheme is constructed from a key agreement scheme as in the "Schemes" document, is the receiver's public key a key transport key or a key agreement key?

p. 36, items 15,16: In some industry practice, ephemeral keys are used more than once, though within a single "session". When Diffie-Hellman is used in S/MIME CMS, the sender generates one ephemeral public key pair per message, and the private key of this key pair is combined separately with each of the recipients' public keys.

pp. 41-42, 5.2.1.2: Can the text on the "manual" method and the "electronic distribution" methods be merged? The only difference is that a CRC is allowed in the manual method.

p. 42, items 1(b) and 2(b): Correctness of a result does not ensure correctness of the keying material (assuming here that "keying material" and "information" are the same in these items). Consider signature verification: an attacker can generate a new public key _after_ a message has been signed, which also correctly verifies the previous signature on that message. (This is the basis for an unknown key-share attack on one version of the Station-to-Station protocol.) To avoid this circumstance, "appropriate cryptographic operation" needs a more precise definition.

p. 43, 5.2.1.3: Please see comments on p. 24 re: confidentiality services and integrity protection. Being "encrypted" only may not be sufficient protection in certain environments.

p. 45, 5.2.2.2, para 1: The disclaimers "Absolute protection ... is not possible" and "(with a very high probability)" are correct, but shouldn't similar caveats also be given for the other protection mechanisms, whether applied in transit or to stored data?

p. 45, 5.2.2.2, item (b): Please see comments on p. 42 re: correctness of keying material.

p. 47, 5.2.3.1: Per item 7, it would be useful to clarify the meaning of "counter" in this context. Per item 9, "modulus" seems unusual as a constituent element of a label.

p. 48, item 1: The restriction "shall not be used to apply cryptographic protection ..." would seem to exclude the use of the key for proof of possession (POP). Since POP is required in order to obtain a certificate (cf. 7.1.5.1.1.2) and the transition to the active state is contingent on certificiate issuance, the restriction on key usage in the pre-activation state may need to be revised.

p. 51, item e: "see Section 7.3.1" --> "but see Section 7.3.1", since that section gives an exception to the statement "do not normally enter the archive state". Perhaps give rationale for this: a key agreement scheme often has forward secrecy, in which case the retention of the private key adds no value.

p. 52, Sec. 7, 5th para (beginning "When functions are combined,"): it's difficult to understand the intent of this text. The example does not appear self-consistent; it would seem that a policy of retaining key material only within a key's cryptoperiod would be well served by removing the key from storage at the end of the cryptoperiod.

p. 54, 7.1.2, last line: suggest changing "IP address validation servers" to "Domain Name System (DNS) Servers", presuming that this more common term matches the intent.

p. 55, 7.1.5.1.1, item 1, 2nd sentence: suggest changing "strength of the cryptographic system" -> "strength of the overall architecture", as the choice between identified and anonymous principals does not impact processing or assurance at the cryptographic layer.

p. 59, 7.1.5.1.2: The requirement that a recipient _immediately_ obtain assurance of validity of an ephmeral public key may be too strong. Some implementations may choose to validate the public key in parallel with its use in the key agreement process. Perhaps say "shall obtain assurance .... prior to relying on that key for subsequent steps in the key agreement process."

However, on a more general note, it is not clear that assurance of validity should be required for ephemeral public keys in all cases. Even with such assurance, the basic ephemeral-ephemeral Diffie-Hellman protocol still needs protection against "man-in-the-middle" attacks, since the MitM can just substitute a new, valid public key. And even without such assurance, the parties have no motivation for providing invalid public keys since only ephemeral private keys are involved in this version of DH.

p. 63, item 1 at top of page: this seems to exclude use of anonymous Diffie-Hellman, which can be an important primitive in some usage contexts.

p. 74, footnote 11: One reason that it can be difficult to "obliterate" a stored secret is that a value, stored for a long time, may "burn into" memory. One way to avoid this is to store shares of the secret rather than the secret itself, and to continuously update the shares. The method is described in

* G. Di Crescenzo, N. Ferguson, R. Impagliazzo, and M. Jakobsson. How to forget a secret. STACS '99. Available via http://www.rsasecurity.com/rsalabs/~markusj.

p. 80, 8.2.5, item 1b: A 3-year upper limit for signature keys may be short, particularly in the case of long, well-protected keys used on an occasional basis at high levels within a PKI.

p. 82-83, item 7, 2nd para: this example assumes that message-level keys will not be retained in wrapped form, and therefore that their wrapping keys need not persist for the useful length of the data. It seems likely that received messages may be stored, indefinitely, in the form that they were received; use of key wrapping should not be restricted in such usage contexts. It may be useful to add an additional balancing example of this form.

p. 84, item 11a, 3rd sentence: This sentence does not appear directly related to cryptoperiod determinations, and particular attacks are mechanism-specific. We suggest dropping the sentence.

p. 87, 8.4, 7th line: per Sec. 7.3.5, note that not all (potential) relying entities may be directly notified in the infrastructure case; it becomes the infrastructure's responsibility to make revocation information available. A similar point applies at p. 93, 8.7.4.

p. 95, 8.8.1 and following: NIST's guidance on "equivalent algorithm strengths" has helped to simplify and clarify the discussion of key sizes. We would like to offer a few additional thoughts to that discussion:

* "Equivalent" is a misnomer, since no proof of "equal value" has been demonstrated. Indeed, as already noted in the guideline, if better algorithms are developed for integer factorization (say), the comparisons could change dramatically. This is particularly the case at the larger security levels. Current research suggests that the algorithm choices and key sizes at the 80-bit security level are likely to offer comparable difficulties, since the methods that will be applied once the cost becomes affordable in the coming decade are likely to be similar to what is already known. In contrast, the algorithm choices and key sizes at, say, the 128-bit level are much harder to compare. With current methods, the cost will not become affordable for several decades (a rough estimate suggests that the cost with current methods would be more than a trillion times greater than that for the 80-bit security level). By then, new methods may well have been developed (e.g. quantum computers) that change the comparisons entirely. For the 192-bit and 256-bit levels are even further removed from practical comparisons today. Instead of "equivalent algorithm strengths", we would suggest one of the following alternatives:

- equal-assurance algorithm strengths
- coordinated algorithm strengths
- associated algorithm strengths

* As a practical example, one might ask under which circumstances one would choose to use a 15360-bit RSA key. If one is concerned that factoring algorithms may improve gradually over time, then this key size may offer a security margin against future developments. It is not clear, however, that this key size offers the same security margin as the corresponding 256-bit symmetric key size. If a quantum computer is developed, then the 15360-bit RSA key may effectively offer no security, whereas the 256-bit symmetric key size may still offer 128-bit security (according to current research). Thus if one is concerned that a quantum computer may be built within the lifetime of the data being protected, one should not use RSA (or other public-key cryptography) at all.

* The primary motivation for the larger security levels, as we see it, is higher assurance. In addition to methods that attack the underlying hard problem such as factoring, there is also the possibility that some information from a secret key may be leaked over time, which is another reason for choosing a larger security level. The larger security levels do provide additional margins of security, but it is difficult to quantify how much. The current selection of algorithm choices and key sizes should be seen as a matter of initial convenience: 80, 112, 128, 192, and 256 are the key sizes for current symmetric algorithms; five sizes for other algorithms need to be chosen; and the sizes for the other algorithms are "derived" by using asymptotic formulas for the difficulty of the currently fastest available methods for breaking those other algorithms. While these correspondences give _lower_ bounds, they are quite different than an algorithm-specific analysis of security margins, which would contemplate potential future developments in methods specific to each algorithm (along the lines of the Lenstra-Verheul tables). Such an analysis would be a challenging exercise, and we do not have any concrete guidance to provide at this point.

* ANS X9.44 includes tables simliar to Tables 8 and 9. Two changes have been made so far which NIST may wish to consider:

- The 80-bit level will be phased out at the year 2010, rather than 2015. This reflects a more conservative assessment of the protection offered at that level. Indeed, if one assumes that today, a "DES cracker" could be built for $10,000 that could recover a 56-bit DES key in 3.5 hours, and that Moore's Law continues for five generations over the rest of the decade, then by the year 2010, an "80-bit cracker" could be built for $10 million that could recover an 80-bit symmetric key in about three months. Recently, Adi Shamir and Eran Tromer published a design for a circuit for the "sieving step" of integer factorization, called TWIRL. They estimate that TWIRL would cost around $10 million and could complete the sieving step for an 1024-bit RSA key in about a year. There is more to factoring than the sieving step, and the estimate depends significantly on unverified predictions about factoring algorithms, but the design underscores the point that the 80-bit / 1024-bit security level is becoming the next (though still distant) target for practical attack.

Given the industry's experience with the slow transition from DES, it would be better to start the transition sooner than later from the 80-bit / 1024-bit security level, as the level will likely linger (and without risk for most data) well into the next decade.

- At the 192-bit level, the RSA key size is recommended to be 8192 bits for interoperability (since 8192 is a multiple of 1024). The minimum of 7680 still applies. A similar recommendation could be made for DSA, D-H and MQV (finite field versions).

p. 96, item 3: It is not clear that random data is required to achieve _maximum_ strength of a hash function. The hash function should be strong even if the input is completely controlled by the attacker. This is the case in typical digital signature schemes, where the message (which is not random) is hashed as the first step of the signature and verification operations.

p. 100, item 3: The security levels for a signature and MAC may be different than for encryption. This is the same point made in Appendix C regarding passwords. An application may require (say) only 112-bit security for MACs, since it limits the number of MAC verification attempts, whereas it requires 128-bit security for encryption. While a conservative design choice may lead to the same security level for the MAC and encryption, it is not clear that this constraint should be required.

p. 101, item c: The final sentence about using different keys for signatures and encryption is good guidance, but out of context for this discussion of key sizes.

p. 101 et seq., Sec. 8.9: This section, describing the contents of a key management specification, seems out of place relative to the preceding subsections of Sec. 8, as it focuses on a specific development process rather than the technical aspects that are emphasized throughout the rest of the document. We suggest that this material be moved to a separate specification.

p. 107, 1st para, last sentence: this text seems to imply a more direct relationship between attack resistance and processing cost than may necessarily be the case; more complex algorithms do not intrinsically provide greater strength.

p. 117, B.3.14.2, note that in some usage contexts IVs may conveniently be stored and retained along with the protected data objects (messages) to which they correspond, so separate archive and/or backup may not be necessary or effective.

p. 122, Appendix C, general comment: it would be worthwhile for this section to discuss the value of password-based key derivation using an "iteration count" as per PKCS #5, as a means to slow guessing trials.

p. 123, Table 10: should provide more emphasis of the fact that the rightmost column isn't a password in the usual sense, but rather a passphrase of several distinct words.

p. 127, C.6, 2nd para: The general concern is that the encryption operation should not have any redundancy, i.e., any ciphertext that is valid for one encryption key should be valid for all encryption keys. This is difficult to achieve, especially if the plaintext already has redundancy (e.g., the plaintext itself is a DES key with parity bits).

p. 128, last para: Some of the large variety of strong password techniques have been constructed so that verifiers need not store values that are plaintext-equivalent to the password presented by the user. In 1st sentence, "inversion" of a password is an unusual term. RSA Laboratories' researchers have been involved in additional research on the problem of deriving strong keys from passwords. We would like to draw attention to two publications, one which will appear later this year:

* W. Ford and B.S. Kaliski Jr. Server-assisted generation of a strong secret from a password. In 9th International Workshops on Enabling Technologies (WET ICE 2000), IEEE Press, 2000. Available via http://www.rsasecurity.com/rsalabs/~bkaliski

* J. Brainard, A. Juels, B.S. Kaliski Jr., and M. Szydlo. A new two-server approach for authentication with short secrets. To be presented at 2003 USENIX Security Symposium.

Editorial Points

p. 12, item 2, "Terms and Acronym" -> "Terms and Acronyms".

p. 16, Ephemeral Keys, "and meets other" -> "and that meet other".

p. 20, Secret key: "not be made public" -> "not made public".

p. 29, 4.2.1, title: -> "Hash Functions" (plural)

p. 26, 3rd para, 1st line: text missing or garbled between 1st and 2nd lines.

p. 32, 4.2.5.1, 2nd line: "ANSI [X9.42]" -> "[ANSI X9.42]"

p. 34, 5.1, 2nd para, 1st line: ".." -> "."

p. 37, Table 1, row 2: center last column

p. 37, Table 1, row 3: "signing private key" -> "private signature key"

p. 37, Table 1, row 6: "private key" -> "private authentication key"

p. 39, Table 1, rows 6,7: extra space before "Integrity"

p. 39, Table 1, row 7: "Private authorization" --> "Private authorization key"

p. 41, Table 2, row 4: In "Security Protection" column, suggest to list "Confidentiality" first for consistency with other entries

p. 41, Table 2, rows 6 and 7: "User" -> "Usage"?

p. 42, 5.2.1, 1st para, 7th line: "is be protected" -> "is protected".

p. 55, 7.1.5.1.1, item 1, 4th line: "owner'" -> "owner's"

p. 66, Table 4, row 12: "presence" -> "its presence"; "available" -> "that is available", for consistency with other entries

p. 83, 1st para, 1st line: "are important" --> "is important"

p. 85, item 15(b), 4th line: missing space before "or"

p. 122, 4th para: "Using a key generated a key" -> "Using a key generated". 5th para: "they, slelect, keep" -> "they select, keep".

NIST's Questions

Page 96, Section 8.8.1, no. 3: Are there any other applications for which collisions are a concern?

* Another application is a commitment scheme, where a provider commits to a message M by providing a hash h(M,R) where R is random, and later "decommits" by revealing R and M. The randomness of R and the one-wayness of the hash function prevents the recipient from determining M before the "decommitment"; but collision-resistance is needed to prevent the provider from decommitting to a different message.

Page 96, Section 8.8.1, no. 4: Are there other applications that do not have collision concerns?

* Other applications may include: hash chains and hash trees; signature schemes where the message is hashed together with a random value, if first-party attacks are not a concern; mask generation functions; and key derivation functions.

Page 99, Section 8.8.2: For 112 bit of security, should the output of SHA-256 be truncated to 224 bits, should truncation not be allowed, or should truncation be optional?

* We recommend the first choice, that the output be truncated to 224 bits. This seems more natural for implementers as it avoids an "exception" to the general rule that the hash value is twice as long as the security level. It also has a slight savings in size, which may help for some applications where the hash size significantly affects the overall transmission (e.g., hash signatures). Truncation would be optional anyway since an application

has the choice of using SHA-256 instead.

* If an application does support both SHA-224 and SHA-256, some care should be taken to ensure that there are no unintended interactions between the two algorithms. That is, it should not be possible to persuade a party that SHA-256 has been used when SHA-224 was intended, or vice versa. The concern can be avoided by requiring that a given key be employed only with one hash function. The same concern would apply to SHA-384 and SHA-512, since the former is a truncation of the latter.

* The point has also been raised regarding ECDSA that the output of the hash function should be _shorter_ than the elliptic curve parameter, to avoid Vaudenay's attack on domain parameter generation. The discussions on this point should be considered in selecting the algorithm and key sizes at the various levels, including 224 bits.

Page 100, Section 8.8.3, no. 4: Are there other combinations to be considered?

* No suggestions here, yet.

Page 122, Appendix C: Appendix C has been provided to invite comment on the use of passwords for the generation of cryptographic keys. Some or all of this material may be provided in another document at a later date.

* As the originator of PKCS #5, which is cited in C.6 as "a de facto industry standard for generating keys from passwords," RSA Laboratories appreciates NIST's attention to the challenges in obtaining keys from passwords, particularly encryption keys. We encourage NIST's more official publication of this material in the appropriate document.

From: Ed Scheidt <eds@tecsec.com>
Date: Tue, 1 Apr 2003 17:26:34 -0500

Within Publication 800-57, Protection Requirements for Cryptographic Information, 5.2.3: Labeling of Cryptographic Information, there are two American National Standards Institute (ANSI) standards that cite a key management architecture where keys are derived from system-specified control information such as labels. The key management scheme is called Constructive Key Management (CKM) and includes many of the elements identified in 5.2.3. In ANSI X9.69 the framework for CKM is identified. In ANSI X9.73 the details are defined for the key combining mechanisms used to generate content encryption keys and key encryption keys.

We recognize that there are two public key management techniques: Key Distro (RSA) and Key Establishment (Diffie Hellman). CKM leverages either key management technique to encrypt objects. As notated in x9.73, CKM may be used with Cryptographic Message Syntax to encrypt a message (as an object) to a set of users sharing a common set of key components. Access to the message content may be controlled by distributing subsets of these key components to users. The key components used for encryption of a specific message are chosen by the sender, and these components define the readership of the message. The sender-chosen components are combined with a random component to produce an object key to be used as the content-encryption key. CKM can also be a symmetric key management as defined in the standards.

It is suggested that reference to the two ANSI standards be included in Pub 800-57 as CKM is a recognized manifestation of a key management scheme that defines how a label may be used to provide information for the use of a key. Also, reference to CKM could be added to 5.2.3.1 as it would be an example of a label practice.

Ed Scheidt
Tecsec Incorporated
eds@Tecsec.com

Comments on:
"NIST Special Publication 800-57:
RECOMMENDATION FOR KEY MANAGEMENT --
Part 1: General Guideline"

David Jablon
Phoenix Technologies Ltd.
April 3, 2003

In the January 2003 draft of NIST SP 800-57 Part 1, it is very encouraging to see the expanded scope of study for key management recommendations, and, in particular, a good start at covering passwords and password-based techniques related to key management. Passwords are ubiquitous in many systems. Yet, there are many issues associated with the use of passwords that have not been adequately addressed in many earlier cryptographic standards, products, and computer security folklore. The proper use of passwords in cryptographic systems is an area where guidance is particularly needed, and where guidance from NIST in particular is especially appropriate. The preliminary discussion in Appendix C of this publication is a good start at focusing on this area.

Most of these comments are in response to the invitation to comment on Appendix C. This section of the publication is a significant and productive step in a process that can, eventually, comprehensively address the appropriate uses of passwords in all cryptographic systems, within a consistent framework of discussion. It is apparent from this draft that the authors saw a need to re-examine commonly held beliefs, and review popular "best practices" in the field, within a larger framework. In some cases, the end result may guide people to question the value of prior held beliefs and the foundation for some commonly-accepted sub-optimal practices. The need for perceptual shift may be due to either the availability of new technology, or due to a broadened perspective on how people actually use cryptographic systems. This guidance will surely encourage the development and deployment of new and better designed systems, as well as a better understanding of how to appropriately use older systems.

These comments include both specific changes to clarify the discussion, and general comments on areas where further guidance can be provided for how to appropriately use passwords in cryptographic systems.

Specific comments

2.1 Glossary

page 19

Non-repudiation

Suggest changing "Non-repudiation" to "Signer non-repudiation" to limit the focus (in particular, to not imply any legal definition) and to match the use in the definition of digital signature.

For the same reasons, a specific suggestion is to change the definition from:

> "A service that is used to provide proof of the integrity ..."

to:

> "A characteristic of processes that provide proof of the integrity ..."

page 21

Shared secret

Suggest changing "Shared Secret" to either "Shared Secret Key", or better still, to "Agreed Key". This would help prevent potential ambiguity and confusion with other common and less precise uses of the term "shared secret", which has often been used to refer to shared keys, or shared passwords, or loosely to refer to values that may be either passwords or keys.

The term "agreed key" more clearly focus on the use in this document, which is specifically to the output of a formally-defined key agreement process, in contrast with the shared keys or other values that may serve as input to an alternative process, such as symmetric encryption.

page 22

Statistically unique

Suggest generalizing this to define it in terms of any arbitrary number of quantities, rather than just powers of 2. For example, some systems may require one to choose randomly and uniformly from a prime-order space of elements, and where, in fact, choosing from a slightly smaller space of 2^n values may be hazardous. The current definition is also somewhat unclear. One suggested rewording is:

> "Statistically unique refers to the characteristic that elements, when chosen uniformly at random from a sufficiently large set, are, for all practical purposes, guaranteed to be unique. An element chosen from a finite set of N elements is said to be statistically unique (with respect to N) if the process that governs the selection of the element provides a guarantee that for any integer L <= N the probability that all of the first L selected elements are different is no smaller than the probability of this happening when the elements are drawn uniformly at random from the set."

page 122

Appendix C - Keys Generated from Passwords

A first comment is that the title of Appendix C is: "Keys Generated from Passwords". This title seems too narrow, and should probably be changed to reflect the broader scope of discussion that exists, and needs to exist, in this document.

One recommended alternative title is "**Key Establishment Using Passwords**". Much of the valuable content of this section describes the limitations of specific methods for generating keys from passwords. In fact, password-generated keys are specifically contrasted with other methods for establishing password-authenticated keys, where the entropy of the key is not derived exclusively from a password. The terms "key establishment", and more specifically "password-based key establishment" are becoming standard ways to encompass the broader categories, as reflected in the works-in-progress of IEEE P1363.2 and ISO/IEC WD 11770-4. The broader category of key establishment further includes both key agreement and key retrieval, and these terms are also consistent with the IEEE 1363 and ISO/IEC definitions.

paragraph 1

Suggest changing:

"Typically, passwords are an authentication and access control mechanism;"

to:

"Passwords are commonly used in authentication and access control mechanisms;".

because passwords are merely data used by a mechanism. They're used in a wide variety of mechanisms that this section separately discusses, including local password verification mechanisms, mechanisms that transmit and verify passwords, challenge/response mechanisms that remotely verify keyed-hashes of passwords without transmitting them, remote password verification mechanisms that have zero-knowledge properties, etc.

Regarding:

> "... But the mechanisms for accomplishing the
> authentication from the password may be cryptographic, and in many cases, the password is effectively serving as a cryptographic key, even a key wrapping key."

The definition of password-generated keys and their limitiations are well-described in this section. However, it seems appropriate for this section to go further, and specifically deprecate or discourage certain practices. In particular, it may be good to discourage the use of password-generated keys (e.g. password-based key wrapping keys) in situations where a key of insufficient entropy would merely provide a false sense of security.

paragraph 2

Recommend changing:

"A key is said to be generated from a password ..."

to:

"A secret key is said to be generated from a password ...".

Some password-authenticated key agreement protocols specifically "generate public keys from a password" in conjunction with an additional random private key. Narrowing the

above sentence to make the definition apply to the case of secret keys eliminates some potential ambiguity.

paragraph 3

Recommend changing

> "Some two party protocols (see Appendix C.7) use a shared password ..."

to

> "Some two party protocols (see Appendix C.7) use a password ..."

In at least two classes of password-based protocols, the Server party does not share password data that is identical to the Client party's data. For example, in augmented password-based key agreement protocols, the server has a public key that is derived from the other party's password-based private key, and in password-based key retrieval protocols, one party has a key that is associated with the other's password but is not derived from the password.

Other suggested changes to clarify this paragraph are highlighted here:

> "Some two party protocols (see Appendix C.7) use a ~~shared~~ password as a part of a cryptographic process that generates a high quality shared key, [where]~~but~~ the key generation process depends on
> conventional random numbers, [and is authenticated by]~~not~~ the password[. In such protocols], ~~and~~ an eavesdropper does not learn anything
> about the password or the key. In this case, the shared key is not generated from the password, [and the password]~~since it~~ is not the primary source of entropy in the key[. H]~~, h~~owever possession of the generated key
> can still be used to authenticate the parties, or to protect an encrypted session."

paragraph 5

The last sentence falls short of providing appropriately specific guidance:

> "... Nevertheless, it is very common to
> generate keys directly from passwords in many applications, and useful protection can sometimes be obtained if passwords are well chosen, systems and applications are properly designed, and, most importantly, users are disciplined in how they, slelect, keep and use their passwords."

If useful protection can be obtained, then one is still left to wonder, specifically, how? This should better summarize the requirements. For example, it seems imprudent to believe that users will be sufficently disciplined to properly use a password with 128 bits of entropy in many applications. It seems better to summarize proper application design issues. The qualities of a "well chosen" password are highly relative to the system in which it is used, and more specific advice is needed to inform the reader on how to "properly design" an application so that passwords are indeed likely to be "well chosen". However, there is no fixed line between "strong" and "weak" passwords that applies to all systems. Thus, useful advice that is not tied to a specific class of system may be impossible to provide.

Suggest changing "it is very common" to "it has been very common". It would seem a desirable goal of these recommendations to make password-generated keys a less common practice, in situations where "proper" user discipline is not likely to be enforced.

Fix spelling error in "slelect".

C.1 Passwords Equivalent to Cryptographic Keys

page 123

Table 10

Suggest that Table 10 be consistent in its treatment of password/key size equivalence. To be conservative, it can be changed to consistently round up, rather than sometimes rounding up and sometimes down.

paragraph 2

Suggest changing "passwords are chosen" to "passwords' components (letters, digits, words, etc.) are chosen" to clarify. A 12-letter word may be chosen "entirely at random", but from a space with far less than 2^56 possibilities.

Suggest deleting "sometimes the key is a little stronger and " in line with comment on Table 10.

paragraph 3

Suggested changes highlighted:

> "Almost anything that is done make the password easier to remember [tends]~~is going~~ to make it weaker,
> that is, to reduce the [number of likely values and thus reduce the]amount of computation needed to attack the password by exhaustion of all
> [likley]~~possible~~ passwords."

page 124

paragraph 2

Suggested minor rewording:

> "[In practice, o]~~O~~ff- line attacks are generally much more [dangerous]~~powerful~~ than on- line attacks. ..."

C.3 Password Strength

The title, introductory statement, and subsequent discussion of section C.3 seems incorrect, or even misleading, unless further qualified.

Regarding:

> It is useful to class passwords into three general categories, by their strength or resistance to attacks, as defined in Table 11: weak, strong and inexhaustible.

Categorization of passwords is only useful relative to a specific mechanism, environment, or application. The specific categorization of passwords by size vs. strength depends on the system. One might be tempted to add a 4-digit ATM password in the "Strong Example" box, but a better solution might be to have different tables for different classes of system.

For example, a 4-digit ATM password is reasonably strong, whereas a 4-digit CHAP or Kerberos password is extremely weak. Why is this so? Yes, the fact that there is a second factor, the ATM card, is one factor. But one must also consider systemic tradeoffs between strength vs. convenience, including the consideration of denial-of-service attacks.

If a definition of "inexhaustible password" is needed, there's a need to say that it must be just like a key, that it needs truly random components, and in particular, not human-chosen or human-selected, which would introduces bias.
There is good discussion of this in C.4. Perhaps a reordering of the material would be helpful to clarify some issues up front before going too far down the path of general advice.

paragraph 4

The paragraph that begins with "Weak passwords are very common, ..." is confusing, since "strength" is a characteristic of a method or overall system, and not of a specific password, unless in the context of a specific system.

page 125

paragraph 2

After the sentence:

> "An eavesdropper who captures the challenge and reply can quickly try all the values of a 4, 5 or 6 digit PIN and determine the PIN."

consider adding:

> "The same is generally also true for typical user passwords, which often have, effectively, only 20 to 30 bits of randomness."

One might also add references to studies on password quality to support estimates of password randomness.

paragraph 4, alternative (1).

Suggest either changing:

> "Obviously, this is cumbersome if good security is provided, but ..."

to:

> "Obviously, this may be cumbersome, but ..."

or else clarify what is meant by "if good security is provided".

paragraph 4, alternative (2).

This practice should **not** be recommended, since it is open to attacks of potentially greater (and unbounded?) efficiency.

paragraph 5

Regarding:

> "Creating and memorizing good inexhaustible passwords requires thought and effort, and is more an art than a science."

Alternative (1) in paragraph 4 has a scientifically sound basis, whereas alternative (2) seems dangerous, and potentially irresponsible.

The subsequent use of the term "well-chosen inexhaustible password" further illustrates the problem. Do we really want to open the door to the concept of "poorly-chosen inexhaustible passwords"?

paragraphs 5 through 8

It is misleading to classify "strong passwords" as a single class for both password-generated keys and other password authentication systems. The recommendation to supplement "weak passwords" with other factors is similarly misleading, without incorporating discussion of denial-of-service attacks.

If this section were to continue to use a fixed terminology for categorizing password strength, then neither "weak passwords", nor "strong passwords", nor "poorly chosen (e.g. human-chosen) inexhaustible passwords" should be used to generate keys.

page 126

C.4 Creating Strong or Inexhaustible Passwords that Can be Remembered

Suggest deleting or rewriting section C.4 in its entirety, in light of the following concerns.

Regarding:

> "Creating long strings that appear to be arbitrary and random, but actually have some structure that facilitates memory, depends on the mind of the password user. There are many techniques that are used, and guidance in this area is often more what not to do than what to do (e.g., don't use the names or nicknames of sports teams, don't use your dog's name, don't use your social security number, etc.)."

Yes, it is common for systems administrators (and even vendors) to provide guidance that tells people what "not to do" in selecting passwords, but this guidance is often extremely ill-advised, and ineffective. Repeating it in this context is of negative value.

When people are told to *not* do something specific, they frequently just do the next worst thing. Advice to "use letters and digits" encourages the single-digit suffix. Advice to not use a dictionary word often results in a dictionary word with a single appended letter or digit. And maybe a lot of people will tend to use the same digit.

In all of this, it is necessary to keep the objective in mind. If the goal is to choose an inexhaustible password, for example, to make, say, Kerberos v4 password-based login cryptographically secure, then such advice almost always fails. If the goal is to choose an appropriate password for a system that appropriately limits exhaustive attack, then much simpler advice may be just fine, and maybe preferrable.

paragraphs 3 and 4

These paragraphs do not provide clear and solid guidance. It would seem better to simply state that human-chosen passwords or phrases are not likely to provide dependable assurance, and that systems designers should strive to create machine-generated manageable passwords or passphrases.

Regarding:

> "But whatever technique is used, the technique usually depends on something well known to the password holder - some favorite text, some obscure interest or hobby, etc. - and then a rule can be applied to produce an apparently arbitrary string."

The problem is that a string may appear to be arbitrary to the user, but not necessarily to an attacker that knows the rule, and knows the person, or more generally, knows human behavior.

C.5 Passwords as Keys in Challenge-Reply Authentication Protocols

Consider changing title from:

> "Challenge-Reply"

to:

> "Hash-based Challenge Reply"

A variety of other protocols, that do not have the limitations discussed in C.5, also use challenges and replies, and, in some cases, appropriately use passwords as keys, internally.

paragraph 5

Suggest changing:

> "Passwords are often used as secrets or keys in a challenge-reply authentication protocol."

to:

> "Passwords have often often been used as secrets or keys in a hash-based challenge-reply authentication protocol, (or worse still, to generate keys, as in Kerberos)."

paragraph 6

Some suggested changes highlighted here:

> "These challenge-reply techniques are sometimes said to use a "one-time password" because the reply is not used again. However, this is a misnomer; the challenge-reply protocol provides protection against the unsophisticated eavesdropper, but does not [protect]~~provide good protection~~ against the [guessing, dictionary, or exhaustive attack by an eavesdropper.]~~more sophisticated attacker.~~ Challenge-reply methods may be seen as more secure than sending a password across a network in an unencrypted message, but are usually vulnerable to eavesdroppers and off- line attacks. [Moral: Don't fool yourself!]"

page 127

paragraph 1

Suggested changes:

> the attacker [may be]~~is~~ quite likely to succeed,[regardless of the strength of]~~however strong~~ the cryptographic functions employed.

C.6 Passwords Used to Generate Encryption Keys

NIST should provide guidance for the practice of passwords used to generate encryption keys. Specifically, NIST should actively discourage the practice of using hashed passwords as keys as a primary or cryptographic defensive technique.

C.7 Strong Keys from Passwords and Servers

Suggest changing title of this section to "Password-authenticated key establishment between two or more parties".
"Keys" should always imply "strong keys". And the traditional roles of Client and Server may vary.

paragraph 5

Suggest changing:

> "This key is not really "generated" from a password, because ..."

to "This key is not said to be generated from a password, because ..."

Suggest deleting the sentence:

> "However, to the user, the key appears to have been generated from the password."

Ideally, the key never appears to the user, and in many such protocols, even if the user did see the key, it would appear to be different each time, even if the password didn't change.

Suggested changes highlighted:

"... the [strength of]authentication of the parties to each other [depends on how]~~is only as strong as~~ the shared
password[, when used in a specific system, resists]~~against~~ an on-line guessing attack."

Suggest deleting:

"Thus, a weak key **should not** be used in this protocol."

In such protocols, the key is authenticated by the password. A very small password may need to be supplemented by other factors, to balance availablity (in the face of DoS attack) with security (the chance of someone getting in with on-line guessing), but does not necessarily need to be supplemented to preserve security.

page 128

paragraph 1

Suggest changing:

"superior to password-based challenge/reply protocols for authentication"

to:

"superior to hash-based challenge/reply protocols for password authentication".

Suggest changing:

generating confidentiality session keys,

to:

generating confidentiality session keys from a password,

paragraph 3

The discussion of [PWENPKI] should state that the technique may require that a prior secure channel be established to the server. An expanded discussion could also discuss the alternatives of sending a password through a server-only-authenticated SSL/TLS tunnel, or sending a password through a dedicated physical link, etc., as well as the limitations of or additional requirements for properly using such approaches.

May also need to add a section (here, or elsewhere, or a reference to such) that discusses the potentially very significant differences between use of TLS/SSL in mutual authentication mode vs. server-only mode. In many cases, TLS may not provide "strong cryptographic authentication of the server" especially when client authentication is not used. The server-only certificate mode of use of TLS/SSL may be especially prone to a various server-spoofing attacks if sufficient countermeasures are not used. Examples of such attacks that may be found in typical web browser application's use of SSL/TLS include:

- DNS spoofing
- user mistakes in typing URL
- user doesn't verify that a redirected URL is correct during a transition from an insecure page to a page that (presumably) uses TLS/SSL.
- user doesn't verify the certificate(s).
- user doesn't verify that TLS/SSL is actually used, and instead relies on visual elements from the rendering of the supposedly secure page.

These attacks are certainly not all applicable to all applications of TLS/SSL in, but a designer should be aware of them. There may also be risks in presuming too much from a certificate-based approach to establishing trust, when there are too many parties that can obtain certificates or too many root authorities, etc.

Suggest adding discussion of the distinctive characteristic of some password protocols that prove knowledge of a password without disclosing it (zero-knowledge password proofs). Specifically, they can eliminate any requirement or dependency for a prior secure channel to the server in guaranteeing security of the password.

The statement that "The key is not truly generated by the password, rather the key is derived from a
conventional random number generation processes." applies broadly to many password-authenticate key establishment techniques that use asymmetric cryptography.

paragraph 6

Suggest deleting the sentence:

> "A disadvantage of the Diffie-Hellman based strong password techniques is that the server must retain the true passwords themselves, rather than an inversion of the passwords (this is also true of password challenge/reply protocols like APOP)."

For example, this is not a "disadvantage" of the class of *augmented* password-authenticated key agreement protocols, of which there are several based on adaptations of Diffie-Hellman.

Suggest changing:

> "These passwords must be strongly protected."

to:

> "These passwords or password-generated data must be strongly protected."

Suggest changing:

> "Nevertheless, when passwords are employed in authentication protocols,"

to:

> "Nevertheless, when passwords are employed in multi-party authentication protocols,"

In general, added discussion of tradeoffs between DoS attacks and small keys would be useful.

page 129

APPENDIX X: References

Suggest supplementing or replacing the reference to the [SPEKE] Internet draft with a reference to relevant cited peer-reviewed publications, such as:

> [Jab96] D. Jablon,
> Strong Password-Only Authenticated Key Exchange
> Computer Communication Review, ACM SIGCOMM, vol. 26, no. 5, pp. 5-26, October 1996.

http://www.integritysciences.com/links.html#Jab96

[Jab97] D. Jablon,
Extended Password Key Exchange Protocols Immune to Dictionary Attacks ,
Proceedings of the Sixth Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET-ICE '97), IEEE Computer Society, June 18-20, 1997, Cambridge, MA, pp. 248-255.
http://www.integritysciences.com/links.html#Jab97

There are also a number of good references listed at http://www.integritysciences.com/links.html for other methods in this general category ... that are not necessarily related to either the company or individual that is providing these comments.

In an expanded discussion, additional references for multi-party password-based techniques would be useful:

[FK00] W. Ford & B. Kaliski,
Server-Assisted Generation of a Strong Secret from a Password,
Proceedings of the IEEE 9th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, NIST, Gaithersburg MD, June 14-16, 2000.
http://www.integritysciences.com/links.html#FK00

[Jab01] D. Jablon,
Password Authentication Using Multiple Servers,
LNCS 2020: Topics in Cryptology -- CT-RSA 2001, April 8-12, 2001 Proceedings, pp. 344-360, 2001, Springer-Verlag.
http://www.integritysciences.com/links.html#Jab01

Reference to [P1363.2] is a more comprehensive and up-to-date reference for some of the above as well as other password-based key establishment methods.

Suggest adding a reference to ISO/IEC WD 11770-4, or an updated alternative for this work in progress.

Entrust Inc. Comments on "Special Publication 800-57 Recommendation for Key Management – Part 1: General Guidance"

April 3, 2003

We would like to thank NIST for the thought and effort that went into this document and acknowledge that it is a valuable contribution in achieving an improved practice of security. It is in that spirit that we present the following comments.

1. Section 3.2

Data integrity is the property that data is not modified in an unauthorized manner without detection. The "without detection" is important, as otherwise MACs and signatures do not provide data integrity. We suggest adding the "without detection" concept to this section.

2. Section 3.5

In order to achieve support for non-repudiation a lot more than simply the use of a digital signature is required. For example, the public key cannot be revoked, the public key must be valid, etc.

3. Section 4.1

The first paragraph states that hash algorithms, symmetric algorithms and asymmetric algorithms are the three classes of approved algorithms. However, the following sections also list random number generators. Thus, RNGs should probably also be added to the list.

Public key algorithms can also be used for encrypting data directly. Since this document does not seem to allow this use of public key encryption, it is worth noting that it is not allowed.

4. Section 5.1

The first sentence of the second paragraph states, "Integrity protection shall be provided for all keying material." Does this include keying material while in operational use? If so, then this should be clarified so that people don't think that a cryptographic integrity protection mechanism is required. Alternatively, the protections provided by the cryptographic module itself and the underlying operating system could provide sufficient integrity protection.

5. Section 5.1.1

Points 17, 18 and 19 discuss symmetric, public and private authorization keys. The distinction between authorization keys and other types of keys seems somewhat artificial at this level. Authorizations keys can be classed as one of the other listed key types (e.g., authentication keys) with the added application-level semantics of providing certain privileges or access rights. Thus, we ask if keys with privileges associated with them have any different requirements than keys without privileges. To us there doesn't seem

to be any substantially different requirements for these keys and therefore wonder if having a separate category for them is warranted. That is, the distinction between authorization keys and others is a higher-level construct, and the properties of the authorization key would be inherited from whatever basic type of key it is.

In Table 1 the Validity Assurance column seems to have 2 definitions and is somewhat confusing. What does "Yes" mean and what does "for association of signing private key" mean? We assume "Yes" means assurance of arithmetic validity and the latter phrase means assurance of possession. All public keys shall have assurance of arithmetic validity and assurance of claimed owner possession of the associated private key, although methods to achieve these assurances may be different for different types of keys. So it is not clear how this column aids a reader's understanding and could be confusing.

6. Section 5.1.1, Table 1

The row for *Public ephemeral key agreement key* states that this key must be protected for integrity until the key agreement process is complete. However, this does not correspond to how ephemeral keys are used in Ephemeral-Static Diffie-Hellman when used in store-and-forward environments, like S/MIME. Normally, the ephemeral public key is not protected for integrity (or otherwise) during transport to the receiving entity or while in storage since there is no requirement to do so. This row in the table should be modified to reflect the fact that this protection is not always required.

However, Section 5.2.1.2, point 1 (b) seems to allow verifying the cryptographic operation (in this case successful decryption) as an appropriate integrity mechanism. If that would apply to this situation, then the linkage should be made clearer.

7. Section 5.1.2, Table 2

The row for *Random number* states that they shall be protected for confidentiality. Random numbers, by themselves, are not always protected for confidentiality (for example, when they are used as a challenge in a challenge-response protocol), nor should they always be. This row should be modified to reflect this fact.

8. Section 5.2.1.2

Point c) in the last paragraph recommends that recipients request that incorrect information be resent as the result of an integrity failure. Often this will open the recipient up to a side channel attack that takes advantage of these types of error messages. Thus, a warning should be given in association with this advice. It may actually be useful to have a section describing appropriate behavior for dealing with errors.

9. Section 5.2.2.3

Another acceptable method of providing confidentiality protection for keying material while it is in storage is to encrypt the material using a key that is then stored in a FIPS 140-2 compliant cryptographic module. This would seem to be allowed by the discussion in Section 7.3.1. Thus, this option should be included in this section.

10. Section 6

During the pre-activation state a key may not be used to process any cryptographically protected information. However, point 1 also states that this is the state of a key during its distribution. It is not possible for both of these statements to be correct as many keys are distributed after they process cryptographic information. For example, key agreement keys will often be used in a cryptographic process before being distributed, symmetric keys will often protect information before being encrypted and distributed to the recipient, and public keys will have to be used to provide proof-of-possession before being certified. Perhaps the statement should be change to "This is the state of a key during its *initial* distribution (i.e., before it's first use)."

Transition 2 recommends that a key pair not be considered active until a certificate has been issued. However, normally in order to issue a certificate a CA will require the key holder to perform cryptographic operations as a proof-of-possession. This would not be allowed if the key were still considered in the pre-activation state. Thus, the description of the pre-activation state should be modified to state that the key should not be used to process operationally protected information, but to allow administrative functions.

The terminology of the "Process-only state" is somewhat confusing, as there are many uses of the word *process* in the document. Suggest instead using "Recipient-functiononly state". Active state allows both originator functions and recipient functions. Recipient-function-only state allows only recipient functions, such as signature verification, MAC verification, or decryption of a message or key. This terminology seems to be more intuitive.

From our point of view, it seems like the deactivated state is really more of a state transition, from either the pre-activation, active or process only states to either the archive or destroyed states. Since keys will not normally remain in a deactivated state without being archived or destroyed, is it necessary to place requirements on this state? Our suggestion then is to remove it.

Transition 12 seems to be missing from Figure 1.

Point b) states "The public signature verification key enters the compromised state only if its integrity becomes suspect." It is also true that it will enter the compromised state if the confidentiality of its private key is suspected of compromised.

Private key transport and private key agreement keys may never enter the deactivated or archive states if they are always required to decrypt encrypted data. A note should be made of this fact.

11. Section 7.1.5.1.1.1

Point 1 states that parameters can be obtained from a self-signed certificate. However doing so leaves one open to certain parameter substitution attacks. The parameter information should be listed as something that cannot be verified from the certificate itself. It might be used as a helper field to select from a list, so use of the field should not

be prohibited, but it should not be used by itself.

Point 2 states that when the CA or RA delegates the verification of a requesting entity's identity to another trusted process, the trust anchor information should be provided along with the unique, unpredictable information. There seems to be no reason why the trust anchor information shouldn't be provided as usual, protected by the unique, unpredictable information.

12. Section 7.1.5.1.1.2

Assurance of possession of the associated private key is also a needed assurance that may be provided by a CA or an RA. This assurance should be added to the first paragraph description of the assurances that can be provided by a CA or RA.

Some methods of obtaining assurance of possession (sometimes called POP) such as use of the public key in a successful execution of the natural function might be compromised if assurance of public key validity is not obtained beforehand. The safe default is to obtain assurance of public key validity before (or concurrently with) obtaining assurance of possession.

13. Section 7.1.5.1.3

In the last paragraph, the assurance of public key validity should be obtained first (that is, before any usage of the public key) and so should be mentioned first.

14. Section 7.1.5.2.1

There seems to be no reason why symmetric keys used only for the storage of information should not be distributed outside an organization if the receiving entity requires access to the information protected by the keys. It is not clear from this description what is meant by an organization. For example, could it simply mean all entities that agree to certain key management requirements? We note that this requirement may have implications for certain disaster recovery scenarios in which keying material may intentionally be stored outside the organization.

15. Section 7.1.5.2.3

Point 1 states that each entity in a key agreement process should know the identity of the other entity. This is not normally the case in store and forward environments, such as S/MIME where the sender of the information can remain anonymous. We note that there currently is no requirement on key transport schemes to identify the message sender. There seems to be no reason why key agreement, in the case of store and forward environments, should be treated any differently than key transport, since they both achieve the same purpose.

16. Section 7.2.2.1

In table 5, it should always be okay to backup a public seed. It might be used for regeneration of (pseudorandom) numbers, for example. Since this value is public, there appears to be no reason to require that it not be backed up.

17. Section 7.2.4

Point 4 states that keys derived from passwords shall be used for authentication purposes and not for encryption. In some situations, this can be a useful technique for protecting information, like private keys, provided the password is of sufficient strength. Annex C would seem to allow such use. It seems that there needs to be some notion of entropy and attack threat of exhaustion included in this section to be able to understand the intent of this prohibition.

18. Section 7.3.1

Table 7 states that passwords should not be archived unless used to detect the use of old passwords. However, it may also be necessary to archive passwords if they are used to generate keys. Another situation that may require the archive of passwords is when servers require their passwords to be securely stored in order to automatically login after an unexpected shutdown.

19. Section 8.1

There is another important reason to not use a key for two purposes, which is the possibility of a bad interaction between the two functions, where one usage weakens the intended security of the other usage.

20. Section 8.2.1

For item 1, another risk factor in determining an appropriate cryptoperiod is the block size, which is especially relevant when considering symmetric ciphers, in particular TDES.

21. Section 8.2.5

It might be worth noting in Point 2 b) that the use of cryptographic timestamping can allow for shorter public key cryptoperiods and thus reduces exposure to cryptanalytic attack.

In item 6, *Symmetric data encryption key cryptoperiod*, it is important for a user to know that the cipher blocksize is a factor. TDES with a blocksize of 64 bits means that information about the plaintext should be expected to start to leak after 2^{32} blocks have been encrypted due to collision concerns and may leak sooner if unlucky. As this concern is probabilistic, NIST should assess what the probability of information leakage should be for government applications and set a limit on the number of blocks encrypted using TDES in either the 2 key or 3 key form. For example, to have less than 1 in a thousand (2^{-10}) chance of any plaintext information leakage, the number of encrypted blocks using a specific TDES key should be capped at 2^{27} blocks, which is 1Gbyte. For AES, the same chance allows 2^{54} blocks to be encrypted, which is effectively unlimited.

NIST should set a maximum number of blocks to be encrypted using a particular TDES key and also give the simple formula so an interested user can determine if the limit should actually be lower for his application. Note that this might result in using AES, as TDES might not meet application requirements, where either algorithm might be acceptable otherwise.

It should also be noted in Point 10 b) that quite often private key transport keys will need to have much longer (essentially infinite) cryptoperiods if access to encrypted data is required.

Points 13 b), 14 b) and 16 b) again seem to be not taking into account store and forward environments, such as S/MIME. In these environments, for example, the private static key agreement key will require a longer cryptoperiod than the public static key agreement key, for the same reasons as for key transport keys. Also, public ephemeral key agreement keys will require a cryptoperiod at least as long as access to the data protected by it is required.

Since not all authorization decisions are made instantaneously, it would seem imprudent to require in Point 19 b) that the public authorization key have the same cryptoperiod as the private authorization key.

A summary table should be provided listing the key type and suggested cryptoperiod. This allows the pattern to be more easily recognized. For example:

Key Type	Cryptoperiod
1. Private Signature key	1-3 years
2. Public Signature key	Tens of years
3. Symmetric Authentication key	<=2 years
4. Private Authentication key	1-2 years
5. Public Authentication key	1-2 years
6. Symmetric Data Encryption key	Session, hour, day, month, tens of years
7. Symmetric Key Wrapping key	Session, day, week, month
8. Symmetric RNG key	Month to 2 years
9. Symmetric Master key	1 year
10. Private Key Transport key	<= 2 years
11. Public Key Transport key	1-2 years
12. Symmetric Key Agreement key	1-2 years
13. Private Static Key Agreement key	1-2 years
14. Public Static Key Agreement key	1-2 years
15. Private Ephemeral Key Agreement key	One key agreement transaction
16. Public Ephemeral Key Agreement key	One key agreement transaction
17. Symmetric Authorization key	<= 2 years
18. Private Authorization key	<= 2 years
19. Public Authorization key	<= 2 years

22. Section 8.2.6

In item 5, the statement that public seeds should **not** be retained longer than needed for validation is true but may be confusing, as the time needed for validation may be a very long time. The purpose of a public seed in domain parameters is to provide assurance to a party, possibly a new party and possibly a neutral third party, that the originating party did not cook the domain parameter generation process in some way to produce a rare

weak result. As such the seed needs to be retained as long as the domain parameters may be used (see Section B.3.14.5), for example, this includes signature verification or recovery of keying material, when means it may be for a long time. Another use for a seed is to reseed a pseudo-random number generator; again, the purpose for keeping the seed is to allow a rerun using the same "random" data, sometimes a rerun may need be done after a very long time. We recommend that this sentence to changed to 'may be deleted' with suitable warnings about the potential to need the seed some time hence. In other words, if in doubt, retain it, as it is public there should be no problem in retaining it, except for the cost of storage, which is minimal.

23. Section 8.4

Since the ability of the sender to create data that will successfully decrypt should not be taken as authentication of the sender, unless the mode of operation provides authentication, the last sentence of point 1 should be modified or removed. It gives the impression that encryption can, in general, be used as an authentication mechanism.

It might be worth noting in the last paragraph of Point 1 that cryptographic timestamping will help protect non-repudiation and authenticity.

24. Section 8.8.1

Key derivation functions are other applications where collisions are not a concern. Of course, they are essentially random number generators, but since they have a somewhat different purpose, it is worth mentioning them separately.

The column for 'hash functions with no collision concern' sticks out from the rest of the table, as it has entries where no other algorithm has entries. The interpretation of this column is difficult since it doesn't really fit with the other columns. Perhaps it would be clearer if the information in this column and the ramifications of choosing a particular hash function in these environments were discussed in a dedicated section.

Table 1 footnote 15 says that the strength of 2 key TDES ranges from 80 to 112 bits, this is actually a simplification. As noted in ANSI X9.52, the strength of 2 key TDES CBC encryption using a random IV is the minimum of (112, 120-n) where n is the base 2 log of the number of known plaintext/ciphertext pairs, if the number of PT/CT pairs is more than 2^{40}, then the strength of 2 key TDES will be less than 80 bits. The simplification will always be true if the number of encrypted blocks is less than 2^{40}, but this should be pointed out as a limitation. In practise, we know that information about the plaintext will start to leak after about 2^{32} blocks have been encrypted. (See comment on Section 8.2.5 item 6 which might obsolete or modify this comment.)

25. Section 8.8.2

There are arguments both for and against the truncating SHA-256 to 224 bits. On the one hand the introduction of yet another hash function with an output size so close to an existing one will add to the complexity of achieving interoperability. This is always the case when more algorithms and options are added to any system and additional OIDs need to be recognized, algorithms implemented (although SHA-224 would be very

similar to SHA-256, it still must be implemented and rolled out to customers before it can actually be used), and protocols modified to handle this additional choice. One then has to ask oneself what the benefit of including a new algorithm is. Thus, industry tends to only support algorithms in sizes that appear to provide clear benefit over shorter lengths. For example, RSA is usually only implemented in keys sizes a multiple of 256, say, and not in smaller multiples even if this might provide a closer security level fit.

On the other hand, if SHA-224 were to be defined as the truncation of SHA-256 to 224 bits and always used in that fashion, this would follow the Jan. 2003 NIST key schemes proposal and recent X9F1 proposal. This may also prevent some subtle interoperability problems when converting between signatures of different types or whenever different hash truncations for the same 112-bit strength may occur. Ambiguity is something to avoid when striving to achieve security.

It is not clear that collisions are always not a concern with random number generation. Consider the case where a user wants to show to a neutral third party that he initialized some deterministic RBGs correctly. If the user can find a collision, he might be able to initialize two deterministic RBGs with the same seed but make it appear that this has happened by negligible change.

26. Section 8.8.3

We believe that the mix and match of hash function sizes with various signature keysizes should be carefully considered to not allow for possibilities that will not be used in practise. The following table may be considered to be included as an interoperability recommendation.

Years	Symmetric Key	Hash-collision	Hash-no collision	DSA	RSA	ECC
Present-2015 (80 bits of security)	2TDES, 3TDES, AES-128	SHA-1	SHA-1	1024/160	1024	160
2016-2035 (112 bits of security)	3TDES, AES-128	One of SHA-224 or SHA-256 depending on outcome of Section 8.8.2 discussion.	One of SHA-224 or SHA-256 depending on outcome of Section 8.8.2 discussion.	2048/224	2048	224
2036+ (128 bits of security)	AES-128	SHA-256	SHA-256	3072/256	3072	256
2036+ (192 bits of security)	AES-192	SHA-384	SHA-384	8192/384	8192	384
2036+ (256 bits of	AES-256	SHA-512	SHA-512	15360/512	15360	512

security)						

Table X Interoperability Keysize Recommendations

27. Section 8.9

This section on the Key Management Specification is difficult to understand. Is this specification supposed to be written by the vendor of the application or by the customer/user? The purpose and use of this document is not clear from the description in this section. Perhaps a more detailed introduction would help to clarify these issues.

28. Section B.3.4

It should also be mentioned (in paragraph 3) that a symmetric key might also be encrypted using a key transport or key agreement key that is being archived.

29. Section B.3.6

Table 1 says that it is allowed to back up or archive random number generation keys in order to re-generate random numbers, contrary to what is stated in this section.

30. Section B.3.10.1 and Section B.3.11.2

Private static key agreement keys and public ephemeral keys should be backed up or archived if they have been used to protect stored data for which access is still required. This may be required, in store and forward environments, like S/MIME.

31. Section C2

The statement that there are “no constraints on an attacker” for an offline attack is too pessimistic and untrue, as there are physical, computation and storage constraints, the same as for a cryptographic key.

32. Section C.6

We are pleased to see NIST exploring the possibility of using passwords to generate encryption keys, where this makes sense. We would like to encourage NIST to pursue this topic in more detail.

33. Section C.7

Similarly, we are pleased that NIST is beginning to consider the use of password authenticated key exchange protocols. We note that P1363 and ISO/IEC JTC1 SC27 are also considering this subject area and would like to encourage NIST to provide guidance in this area.

From: "Walker, Jesse" <jesse.walker@intel.com>
To: GuidelineComments@nist.gov
Date: Thu, 3 Apr 2003 08:52:47 -0800

This is already a nice document, so my comments are limited.

The documents contains numerous references to SP 800-56. Clause 5.3 of SP 800-56 defines key derivation functions. The key derivation functions defined in Clause 5.3 of SP 800-56 are fine, but the list of approved key derivation functions seems deficient. Here are two suggested additions for you to entertain:

a. In order to be compatible with commercial standards such as IPsec, it would be useful if an HMAC-SHA based key derivation function were also approved. This would be similar to the "3.5.1 concatenation key derivation function (Default)":

1. initiate a 32-bit big-Endian counter as 0x00000001
2. j = ceiling(keydatalen/hashlen)
3. for i = 1 to j by 1, do the following:
    3.1. compute Hash[i] = HMAC-H(Z, counter || U || V || [SharedInfo])
    3.2. increment counter
4. let Hhash = Hash[j] if (keydatalen/hashlen) is an integer, otherwise let
    Hhash = (keydatalen - (hashlen * (j-1)) leftmost bits of Hash[j]
5. set DerivedKeyMaterial = Hash[1] || Hash[2] || ... || Hash[j-1] || Hhash

where Z, U, V, and SharedInfo are as in the original. The allowed values for H would be one of the allowed values from Cluase 5.6 of SP 800-56.

b. To facilitate implementations in some constrained environments, it would be useful if a key derivation function based on AES-CBC-MAC were defined as well. The suggested key derivation function would be similar:

1. initiate a 32-bit big-Endian counter as 0x00000001
2. j = ceiling(keydatalen/hashlen)
3. for i = 1 to j by 1, do the following:
    3.1. compute Hash[i] = AES-CBC-MAC(Z, counter || U || V || [SharedInfo])
    3.2. increment counter
4. let Hhash = Hash[j] if (keydatalen/hashlen) is an integer, otherwise let
    Hhash = (keydatalen - (hashlen * (j-1)) leftmost bits of Hash[j]
5. set DerivedKeyMaterial = Hash[1] || Hash[2] || ... || Hash[j-1] || Hhash

where again Z, U, V, and SharedInfo are as in the original, except Z must be constrained to 128, 196, or 256 bits in length.

Jesse Walker
Intel Corporation
JF3-206
2111 N.E. 25th Avenue
Hillsboro, OR 97124
(502) 712-1849
jesse.walker@intel.com

From: "Joshua E. Hill" <jhill@infogard.com>
Date: Fri, 4 Apr 2003 16:26:47 -0800

To whom it may concern:

These comments pertain to the draft SP800-57, Part 1, dated January 2003.

Introduction: "All keys need to be protected against modification, and secret and private keys need to be protected against unauthorized disclosure." It would seem that all keys should be protected against unauthorized substitution and modification.

In the Glossary:
The definition for *Hash-based message authentication code (HMAC)* should probably refer to FIPS 198 given that HMAC generally refers to a specific way of using a cryptographic hash to create a MAC.

The definition for "Hash function" is really a definition of a "Cryptographic hash function". "Collision Free" should read "Collision Resistant".

The definition of "Key Transport" includes the statement "When used in conjunction with a symmetric algorithm, key transport is known as key wrapping." I don't believe that this is correct, as keys can be wrapped and then not transported. Further, the definition of "Key Wrapping" provided later does not state that Key Wrapping is necessarily included as a portion of Key Transport.

Within the definition of "Private Key", the described uses of a Private Key appear to be not wholly correct. The private key is not always used in creation of the public key (RSA key generation, for example, uses a public key component in the creation of the private key). The statement "Compute a piece of common shared data, together with other information" seems needlessly cryptic, and should probably be revised. It seems likely that this statement was included to include public key agreement algorithms (e.g. Diffie-Hellman), but this isn't really clear.

Within the definition of Pseudorandom Number Generator (PRNG), the definition of "cryptographic pseudorandom number generator" does not make reference to the idea that the seed should be difficult to guess, given the cryptographic PRNG's output.

Section 4.2.6 states "Pseudorandom number generation (PRNG) algorithms defined in [FIPS 186-2] (including those defined in [ANSI X9.31] and [ANSI X9.62]) use hash functions to generate random numbers." This is misleading, as the PRNG specified in ANSI X9.31 uses TDES and the PRNG specified in FIPS 186-2 can be constructed using a DES based 'G' function.

In Section 5.1.2, it is unclear how the statement "Other information used in conjunction with cryptographic algorithms and keys also **should** be protected." interacts with the

various "**shall**" requirements imposed on parameters (which would qualify as "other information used in conjunction with cryptographic algorithms and keys" in the numbered list following this statement). It should be made clear whether the requirements marked as "**shall**" are binding in all cases or only in cases where the decision has been made to protect information used in conjunction with cryptographic algorithms and keys.

In Section 5.1.2 the statement is made, "In this case, intermediate results **shall not** exist outside the cryptographic boundary of the cryptomodule. If a FIPS 140-2 Approved cryptomodule is being used, then protection of the intermediate results is taken care of by the cryptomodule." These two statements seem to be contradictory. Perhaps the first sentence should read "In this case, intermediate results **shall not** exist outside the cryptographic boundary of the cryptomodule in unprotected form."

Section 5.2.1 states, "This may be accomplished using “manual methods” (i.e., the information is in hard copy or on an electronic media, such as a CD-ROM), or using electronic communication protocols." The parenthetical "i.e." should be "e.g." unless this list is intended to be a comprehensive list of the appropriate mechanisms. Further, parallel construction suggests that you identify the option other than "manual methods" as "automated methods" (for consistency with earlier terminology and FIPS 140-2), and then include the provided example. The term "electronic distribution via communication protocols" should probably be written as "automated establishment" throughout the document.

Section 5.2.2.3 seems to suggest that storing encrypted (perhaps integrity protected) keys external to FIPS modules (and without the aid of some physical security provided by a safe) is inappropriate. This should be allowed, provided that the keys are protected with an appropriately strong mechanism.

In Section 6 it would seem that if a key is compromised before being archived, it cannot then be archived. This seems to contradict earlier statements about both compromised and deactivated keys existing in the key archive. In addition, it would seem that a key could be compromised while in the archive state. As such, it would seem that there should be transitions between the "Archive" and "Compromised" states. If this is changed, the description for Transition 7 should be revisited.

The definition of "Key Establishment" provided in the first paragraph of Section 7.1.5 seems to contradict the definition in the glossary of this document, as well as the definition used in FIPS 140-2 in that it describes "Key Generation" as a sub-component of "Key Establishment". Key Generation should be addressed in a separate section from Key Establishment.

In Section 7.1.5.1.3, the requirement is imposed on split knowledge procedures that "the key **shall** be split into multiple key components that are the same length as the original key;” It is unclear why only ideal secret sharing systems should be allowed. There is an equivalent requirement imposed on symmetric keys in Section 7.1.5.2.1.

Section 7.1.5.2 does not allow for symmetric keys that are generated and then **not** distributed. This is a common occurrence, and it ought to be explicitly allowed. (Further, this use is mentioned in Section 7.1.5.2.1.)

Sections 7.2.4 (key derivation techniques) and 7.2.3.2 (key update procedures) describe methods for key generation that may be inappropriate within FIPS 140-2 modules. FIPS 140-2 states, "Cryptographic keys generated by the cryptographic module for use by an Approved algorithm or security function shall be generated using an Approved key generation method. Approved key generation methods are listed in Annex C to this standard." Annex C to the FIPS 140-2 standard lists approved random number generators, but no key derivation or key update methods.

Sections 5.1.2 and 7.1.5.3.3 prohibit even the secure distribution of shared secrets. This precludes sharing state among trusted modules for the purpose of fault tolerance and/or shortened key exchanges (e.g. TLS session resumes). This restriction should not be undertaken lightly, as it would restrict a useful technique for many environments.

In Section 8.8.1, Number 8 asks if there are any other cases where collisions are not a concern in hash strength (after citing PRNGs as a possible example). This should probably be phrased not in terms of "collisions", which are often interesting, but in the applicability of the birthday attack on any of these particular uses of a hash function. Any use of a hash function as a PRF (as in IPSec or TLS, for example) does not expose the hash primitive to a birthday attack.

In Section 8.8.1, the claimed strength of two key triple DES is 80 bits of security. There is a chosen plaintext attack against two key triple DES that requires on the order of 2^56 operations and 2^57 storage. It is only when the attack is made known plaintext (rather than chosen plaintext) that the attack requires on the order of 2^80 operations with 2^40 known plaintext/ciphertext pairs. Further, if more than 2^40 pairs are known, the resistance to attack falls below 2^80. (See Fact 7.40 in the HAC. "If t known plaintext-ciphertext pairs are available, and attack on two-key triple-DES requires O(t) space and 2^(120-lgt) operations". So, for instance, if the attacker has 2^80 known plaintext/ciphertext pairs, the attack takes only 2^40 operations).

Joshua Hill
InfoGard Laboratories

From: "Ellison, Carl M" <carl.m.ellison@intel.com>
Date: Fri, 2 May 2003 14:30:48 -0700

In Part I: sec 3.5 - the wording is better than usual, but leaves open the real issue with non-repudiation. Since a person does not do cryptography in his/her head, there must be an intermediate device (computer, in our cases) applying the signature key. Since it is not possible to prove that the computer involved behaved honestly or that the person to whom a key was assigned was actually in control of that computer, it is not possible to prove that any particular person was responsible for a particular digital signature. In the general case (not using specially protected computers with human witnesses of the act of making a signature), my conclusion is that non-repudiation is not achievable.

- Carl

+---+
|Carl Ellison Intel R & D E: cme@jf.intel.com |
|2111 NE 25th Ave T: +1-503-264-2900 |
|Hillsboro OR 97124 F: +1-503-264-6225 |
|PGP Key ID: 0xFE5AF240 |
| 1FDB 2770 08D7 8540 E157 AAB4 CC6A 0466 FE5A F240 |
+---+