Public Comments Received on Draft NIST SP 800-133 Revision 2: Recommendation for Cryptographic Key Generation

(April 17, 2020 deadline)

Brown, Dan

From: Dan Brown (danibrown@blackberry.com)
Date: April 17, 2020

Editorial suggestions for SP 800-133r2

Dan Brown

Overlap with SP 800-90C? Didn't 800-90C specify combining RBGs using XOR? Is 800-133 the same idea, except that one input to the XOR, say V, need not be an approved RBG? Maybe explain this?

> NIST: 800-90C has been extensively modified and the latest draft has not been provided for comment yet. The latest draft does not include combining RBGs.

Example for V: Include as an example for V, the output of /dev/urandom or similar practical kernel random number generators. (Unless you intend /dev/urandom to be used as an entropy input for SP 800-90, instead of as V, or if don't like /dev/urandom so much that you don't want it near a key.)

> NIST: No change. I think what you are thinking of is a 90C topic.

Length of V: Mention that V could be variable length, like U, for example, in the interactive calls to RBG from in rejection sampling from FIPS 186.

> NIST: V is mentioned in Section 4. Since it is XORed with U, it has to be the same length as U (i.e., *bLen* bits), This is stared in line 101.

Override other NIST documents: FIPS 186-4 and SP 800-90 already directly combine to generate keys: FIPS 186-5 says to use an RBG from SP 800-90. But SP 800-133r2 overrides this, introducing an extra modification of xor-V, as an exemption to the literal reading of the other documents. Overriding other NIST algorithms, should be duly emphasized, and expressly stated. Perhaps, the affected documents should be revised to accommodate the 800-133 exemptions

> NIST: 800-133 is an alternative and not intended to override the other documents. The CMVP follows SP 800-133 in their IGs. IG D.12 states that the methods in the relevant sections of SP 800-133 can be used to generate a symmetric key or a seed for generating the asymmetric keys, thus making it clear that an output of SP 800-90A DRBG does not need to be used *directly*.

Clarify what "keys" are, or are not: Do they include ECDSA per-message secrets, block cipher mode nonces, or initial values, user-selected passwords? Under a quick, literal reading, I cannot tell if these are intended. For example, I once tried to trace through all the definition, I found that a user-selected

password is an input to a key derivation function, and that an input to a key-derivation is a key. Am I to conclude is a user-selected password is also a key? I don't think that is the intent.

NIST: The definition of cryptographic keys now includes the following statement: "The specification of a cryptographic algorithm (as employed in a particular application) typically declares which of its parameters are keys." Sections 5 and 6 describe the sorts of keys that this document addresses, and points to specific documents or further instructions when necessary.

The generation of key pairs for ECDSA is addressed in FIPS 186, as discussed in Section 5.1 of 800-133. FIPS 186 has always referred to secret numbers for both DSA and ECDSA. However, the following statement has been inserted "(which includes the generation of "secret numbers" used as ephemeral private keys)."

Regarding key derivation: The one-step KDFs in SP 800-56C do not require anything called a "key" as part of their input. The input to those KDFs (and those of SP 800-108) may include many things that are definitely not keys. The "key" in "key derivation" refers to the output, not the input. Nowhere in SP 800-132 is the input password/phrase called a "key"; but the output (derived from the password/phrase) is explicitly called a "master key."

Clarify what an "algorithm" is: At one point an algorithm is said to define a mathematical function, which suggest an algorithm is deterministic under a literal reading, but that might not be intent.

NIST: The definition for algorithm that is used in several other documents (e.g., 800-56B and 800-90A) was added to the glossary: "A clearly specified mathematical process for computation; a set of rules that, if followed, will give a prescribed result."

Glossary location and length: Move to end [see Knuth's AoP, for example], and shorten. In a direct reading, definitions should appear in logical order, not alphabetical order.

NIST: No change.

Fix clash about "parameter": In ECC, a "parameter" is a value that is usually system-wide-fixed, such as the curve equation, base point, but in SP 800-133, a "parameter" is a run-time-variable input.

NIST: The definition for "parameter" that is used in 800-152 was added to 800-133: " A value that is used to control the operation of a function or that is used by a function to compute one or more outputs." This definition covers both meanings.

Drop redundant definitions, especially data integrity, origin authentication, and entity authentication: The security goals of the cryptographic algorithms using the key generated in SP 800-133 should already be handled adequately in each document describing the cryptography algorithms. For example, FIPS 186 can say what the security goals of a digital signature are: it seems redundant for SP 800-133 to repeat the security goals for a digital signature. Instead, 800-133 should only define security goals specific to keys and key generation, such as the secrecy of the keys as measured by entropy.

NIST: No change. The intent of this document is as a general key-generation document, referring to other documents when a subject is discussed elsewhere. However, context needs to be included, along with definitions for the context information.

Also, some of the definitions seem too informal, and might contradict past definitions. For example, I had thought entity authentication precludes replay attacks, so does not apply to digital signatures, or any other single-pass communication, but 800-133 says signatures provide entity authentication. Under this definition, entity authentication and origin authentication almost entirely overlap. Dropping the definitions should fix this.

NIST: Digital signatures are used for several purposes. The terms "source authentication" and "identity authentication" are used in 800-57 Part 1, so 800-133 has been changed to use those terms (origin authentication and entity authentication are indicated as alternative terms). The intent of source authentication is to provide assurance about who sent a document. The intent of identity authentication is to provide assurance of who you are (perhaps in response to a challenge). The difference is arguably slight and is discussed in 800-57 Part 1.

Repair glossary definition of "key": The glossary seems to imply that an input to an algorithm is a key if the output depends on that input. For example, in T=HMAC(K,M), the input M would a "key" in the glossary's sense because T cannot be computed without knowing M.

It is not easy to define a concept as a general as "key". (The terminology "public key" makes it even more difficult.)

In formal analysis, one or more inputs are *arbitrarily* labelled as keys, then security is defined under the condition adversary does not know the key. That approach might work in standards. For SP 800-133, you could say that a key is any input declared as a key by the document specifying the cryptographic algorithm.

NIST: The definition of key has been used in many documents. However, the following was added to the definition: "The specification of a cryptographic algorithm (as employed in a particular application) typically declares which of its parameters are keys."

The 800-133 definition seems to try reverse the formal approach, by *deducing* which inputs should be labelled as keys. Logically, this seems to presume a notion of security, and then derive what inputs are keys, by how they affect security.

This reverse approach seems harder than the formal declarative approach. I am not sure how to formally define security without first formally defining keys. For example, how can the security aim of HMAC be stated, without stating which inputs are kept secret from the adversary?

Nonetheless, security somehow seems more fundamental than keys: secret keys are a tool used to achieve the task security. It seems worthwhile to define the task first, and then define the tools. So, maybe this reverse approach is worthwhile, after all.

Perhaps, (secret or private) keys could be re-defined in this manner as

*a **key** is an input to cryptographic algorithm whose secrecy from an adversary (as measured by entropy) is key (pun intended, replace by "essential") to the security of a cryptographic algorithm.*

Returning to the example of HMAC, the adversary generally knows, or even chooses, the input message M. Therefore, M is not a key of HMAC, under the re-definition above. For an encryption scheme, such as AES-CBC or RSA-OAEP, the adversary may again know or choose M, but the adversary still cannot distinguishe whether a ciphertext C is an encryption of M, or completely random value (in the space of all ciphertexts).

One problem with the re-definition above is that some types of security may also rely on the secrecy of the message. In this case, we can consider M to have a role of a key, and this can also be quantified by how secret M is. (Perhaps some application of HMAC, maybe HMAC-DRBG or HKDF use M as a "key", under the re-definition above.) For SP 800-90C, you probably do not want to take the view of partially secret inputs as keys, since the main contribution of 800-133 is about XORing an approved RBG output U with an arbitrary value V.

Another possible problem with the re-definition above is that advanced types of security address what happens when a key is exposed. But in these advanced types of security, other inputs serve as keys, so maybe the re-definition survives this problem.

As said at the outset of the re-definition above, it does not address public keys. It is therefore severely limited. It is unclear how to define a public key deductively from the security of a cryptographic algorithm. Behind every public key, there is a private key, which must meet the definition above, and there is a one-way function between the two (some of the glossary definitions already cover this), so maybe that is the solution. (Is then a hash a symmetric key a public key?) Often, there is some extra security attached to a public key, namely its association with an entity.

Powers, Mike

From: Michael.C.Powers@leidos.com
Date: March 5, 2020

Upon reviewing the draft NIST SP 800-133 Rev. 2 document, I note that lines 400 and 401 designate the restrictions on a '*salt*' value to be used in the context of combining multiple keys via HMAC. What is unclear to me currently based on these two lines in the document is if there are any restrictions on its length. I would suggest, perhaps, in a similar fashion to what you do for the 'n' and 'm' values in the following lines that you would include a statement along the lines of "This method requires that the *salt* length ≥ *<some value>*."

This would clarify if the salt can be length 0, or if it must be a positive length (such as >= 1, or >= 112 bits, etc.)

Thanks,

Mike Powers
Leidos CSTL Technical Director
AT&E Labs | Commercial Cybersecurity | Phone: 443.367.7422

NIST: HMAC specifies the handling of the input key, allowing it to be all zeroes when a null string is provided as the key. A clarification has been inserted into Section 6.3 in note b under method 3.

Umesawa, Kentaro

From: kentaro.umesawa@kioxia.com
Date: March 23, 2020

Dear SP800-133 authors,

My comments:

Is this typo in "Appendix A: Revisions" ?
"Changes #7" says "Section 6: Added a 4th bullet about deriving a …", but this seems to point at 4th bullet of Section 6.2.2.

NIST: Listed the change as being in Section 6.2.2 and corrected the change to address concatenating multiple keys.