Public Comments Received on Draft NIST SP 800-56C Revision 2: Recommendation for Key-Derivation Methods in Key-Establishment Schemes

(May 15, 2020 deadline)

Dan Brown

From: Dan Brown" <danibrown@blackberry.com>

Date: 5-15-20

Hybrid support

BlackBerry commends the Revision 2 modifications to 800-56C that aim to support hybrid combinations of key-establishment scheme. These hybrid modification help resist potential weaknesses in one key-establishment.

The most looming risk of potential weakness is from quantum computer attacks on (elliptic curve) Diffie-Hellman key agreement (and related MQV), and on RSA key transport. (Other risks are more remote, but perhaps also worth mitigating.)

When NIST completes its PQC project, being ready now for hybrid, will integration of PQC with ECC easier.

Reducing options

We propose reducing the number of options in 800-56C, because it will help

- improve interoperability,
- shorten 800-56C,
- make implementation easier,
- make testing easier,
- make security analysis easier.

NIST response: NIST tried the one (or two)-size-fits-all approach in its initial SP-800-56 documents and was criticized for disallowing everyone's favorite method. No one is forced to support every flavor of key derivation, but there are a large number of defensible "legacy" versions of key derivation still in use.

> Likely, all the current options are secure, so it may well be difficult to choose which to keep.
>
> Some options in 800-56C are inherited from SP 800-108, Recommendation for Key Derivation Using Pseudorandom Functions (Revised), and from SP 800-135,

Recommendation for Existing Application-Specific Key Derivation, so reducing options in 800-56C may require reducing options in these other documents (when it is time to revise these).

Backward interoperability and standards alignment

Recent discussion on the TLS and CFRG mailing lists have raised the issue of whether SP 800- 56C would be compatible with TLS 1.3, which uses HKDF extensively. By comparison, SP 800-56C, and the document SP 800-108 it refers to, require very thorough interpretation to determine HKDF compatibility.

NIST response: See the notes on Acumen Security's feedback. TLS 1.3's has no SP 800-56C issues. (However, some finessing may be required for the SP 800-108 and SP 800-133 requirements.)

Other NIST documents already recommend TLS 1.3, indicating NIST intends to support TLS 1.3. Perhaps the best way to achieve the general compatibility between NIST and IETF might be best achieved using SP 800-135 Recommendation for Existing Application-Specific Key Derivation Functions, instead of re-specifying HKDF through SP 800-56C and SP 800-108.

Ease of key derivation

Many different alternatives for key derivation seem to be secure.
This situation is quite different from the that of more fundamental cryptographic primitives (block ciphers, key establishment primitives, and so on), where finding secure examples is difficult.

The most serious potential modes of failure in designing key derivation schemes seem to be

- deriving a guessable key by losing (pseudo)entropy,
- accidentally generating the same key when different keys are needed, and
- creating bias in keys (fixed bits in keys, or two related keys).

With hash functions or block ciphers, most of these defects can be avoided quite easily, which probably explains why so many alternatives have been proposed.

Even without hash functions, quite good key derivation is possible, provided that the threat of related keys is mitigated. For example, in HMAC the inner and outer key differ only by an XOR operation. In SP 800-133r2, Recommendation for Cryptographic Key Generation, keys generated from random numbers can be combined using an XOR operation.

Advanced key derivation

Theoretical security properties of key derivation have been studied by H. Krawczyk, Y. Dodis, and others.

It appears that they have greatly expanded the scope of key derivation, beyond how it is used in key establishment, to the point where it overlaps significantly with other cryptographic areas such as pseudorandom bit generation, which is covered by a NIST DRBG.

NIST response: This is probably true; there is a fine line (that is often hard to see) between the requirements for key-derivation methods and pseudorandom bit generation. Not too surprising, since it's hard to imagine a case where you wouldn't want the derived keying material to look as much as possible like a string of bits that were selected uniformly (and independently) at random.

This overlap means that the more advanced forms of key derivation functions in SP 800-56C, the two-step functions like HKDF, do extra work beyond what it is needed in key establishment, as in SP 800-56C. The HKDF-like functions include mitigations designed for other applications, almost DRBG-like mitigations.

Using these advanced key derivations in SP 800-56C should not hurt the security of key establishment. This situation does provide an opportunity for the narrowing the list of options, to the extent that some of the advanced security features, such as the options in SP 800-108, are redundant for SP 800-56C, and could perhaps be excluded from SP 800-56C.

Researcher Criticism of HMAC Security

Other researchers, such as Koblitz and Menezes, have criticized some of the provable security claims made about HMAC. These criticisms might be worth considering if they extend to the HMAC-based two-step key derivation functions (the HKDF-like).

Dodis and colleagues have pointed out the potential insecurity of using CBC for key derivation. We have not reviewed this option or the claims of insecurity, but regard these researchers as reputable, so their claims should be carefully considered.

NIST response: This is all worth looking into but there doesn't appear to be an imminent threat.

Modularity

Previous versions of 800-56A, Recommendation for Pair-Wise Key-Establishment Schemes Using Discrete Logarithm Cryptography, specified key derivation, but the current version refers to key derivation via SP 800-56C.

Such references between the specifications make them more modular, which has many advantages such flexibility and an almost microscopic focus. However, a

possible disadvantage is to make the specifications less self-contained, and to lose sight to the big picture, and the overall system.

We recommend being careful not to continue too far down the road of modularity.

NIST response: Point taken. With the SP-800-56 series, there was an incentive to reduce what would at best have been redundancy between the treatment of key derivation in SP 800-56A and SP 800-56B, but in practice had become a problem of inconsistency, as the update of one document sometimes put it out of sync with the other document.

Phillip Hallam-Baker

From: Phillip Hallam-Baker <phill@hallambaker.com>
Date: Monday, May 11, 2020

I don't support making HKDF a NIST standard in its current form because there is a flaw.

Consider the case in which the initial keying material is formed by concatenating two items, the second of which is a variable length string.

As currently specified, the values for o_key and i_key are the same for a key k and that same key with a zero byte concatenated. This might not seem like a big deal but the whole point of defining common building blocks for crypto is to eliminate all the bear traps we can.

The problem can be eliminated as follows:

1) Generate i_key by padding the key value with byte x before XORing with the i_mask

2) Generate o_key by padding the key value with byte y (y <> x) before XORing with the o_mask

This ensures that the values of o_key and i_key will change if zeros are appended.

NIST response: The fact that HMAC treats input keys K and $K||000...0$ as equal (as long as they are both less than one input block long) is not an issue when a fixed key length is specified (and enforced by users).

The only case where an HMAC key is used by SP 800-56C as a key-derivation key (rather than as a possibly public salt) occurs in the two-step key-derivation methods if the PRF-based KDF (from SP 800-108) utilizes HMAC as its PRF.

But in such cases, the (ephemeral) key-derivation key (which is used as the HMAC key) is of a known, fixed length – it is the output of the extraction process in the first step (using the same HMAC function as used by the KDF) and its length is unambiguously determined by the (untruncated) length of the underlying hash function's output.

In an HMAC-based extraction step of a two-step KDM or an HMAC-based one-step KDF, a possibly public salt is used in place of the HMAC key. Since SP 800-56C does not rely on those salt values in its security-related guidance, it is not clear what the security ramifications might be if there was some confusion between the use of *Salt* and *Salt* ||000. . .0 (which will not affect the output of the extraction or key-derivation processes).

The participants in a key-agreement transaction utilizing a "salted HMAC" are required to have either agreed upon the salt value or defaulted to using a fixed length all-zero bit string (of a length equal to one input block for the underlying hash function). If by some fluke *Salt* were used by the parties in one key-agreement transaction and *Salt*||000. . .0 were used by the parties in another, what would be the harm? The independence of the keying material derived in the two transactions is a function of the other inputs, including the shared secrets formed during the transactions, which are used in the "message slot" of the HMAC. Both SP 800-56A and SP 800-56B discuss the steps that participants in a key-agreement transaction might take to obtain assurance of the "freshness" of the derived keying material.

Ryan Thomas

From: <rthomas@acumensecurity.net>

Date: Friday, May 15, 2020

Thank-you for the opportunity to provide comments on this draft revision of SP 800-56C. Please accept Acumen Security's (an Accredited Cryptographic Security Testing Lab under NVLAP) feedback/comments. In addition, we have been asked to submit a comment on behalf of Google's GCP (Google Cloud Platform) team:

Acumen Security Feedback:

Our understanding is that this draft does not explicitly include the standalone HKDF extraction step option utilized in TLS 1.3 (though we see that RFC 5869 is cited). Over time, as SP 800-131A algorithm transitions remove options in earlier TLS versions (such as the RSA key transport using PKCS #1 v1.5), it will be extremely important to have a clear and straight-forward way to validate modern protocol versions under FIPS 140-2/3. TLS 1.3 is being widely adopted industry wide, it is critical that the standalone HKDF is NIST Approved in a NIST recommendation very soon. This way, modules/products can confidently move forward with TLS 1.3 in FIPS module validations.

We understand from a recent thread on the IETF TLS mailing list that Section 5 in SP 800-108 states "*Alternative orders for the input data fields may be used for different KDFs.*" . It also states "*One or more of these fixed input data fields may be omitted unless required for certain purposes as discussed in Section 7.5 and Section 7.6*.". After an extraction step, the output is a pseudorandom key. In an email to the mailing list, NIST CTG's Quynh Dang goes on to state that the purpose of any of these KDFs in SP 800-108 is the same as the purpose of the expansion step. Therefore, they are allowed for being used as expansion steps.

Section 6.3, item# 3 in draft version 2 of SP 800-133 also seems to specify an option for HKDF's extraction step that is very similar to the external pre-shared key or a resumption in TLS 1.3.

Finally, Section 3.3.3 in SP 800-52rev2 explicitly lists the HMAC-based Extract-and-Expand Key Derivation Function (HKDF) described in RFC 5869 as "*Validated Cryptography". This section states "All cryptographic algorithms that are included in the configured cipher suites and the random number generator shall be within the scope of the validation*" implies that it should be possible to obtain a validation or to vendor affirm this HKDF.

While there seems to be indications in various NIST recommendations that the HKDF in TLS 1.3 is analogous to allowed/Approved KDFs, from what we can ascertain, the standalone HKDF in TLS 1.3 is not explicitly allowed/approved yet.

There is currently a lot of confusion as to the status of the HKDF in TLS 1.3 as it relates to NIST recommendations. We appeal to NIST to please provide clear direction and clarification on the

HKDF in TLS 1.3 (ideally in one publication). We need this HKDF officially approved for key exchange as soon as possible so that validation testing (ACVTS and FIPS 140-2/3 validation) can move forward.

Thank-you for your consideration,

Regards,
Ryan Thomas

NIST response: SP 800-56C specifies the key-derivation methods that can be employed, as needed, to derive keying material from a shared secret *Z* that was generated during the execution of a key-establishment scheme specified in SP 800-56A or SP 800-56B. Its scope does not extend to methods used in other (perhaps more general-purpose) applications of key derivation.

In TLS 1.3, any SP 800-56A-style shared secret *Z* enters as the "(EC)DHE" value in the portion of the "key schedule" shown below:

```
                [Derived Secret Salt from previous steps]
                    |
                    v
(EC)DHE -> HKDF-Extract = Handshake Secret  [ = HMAC(Salt, (EC)DHE) ]
                    |
                    +-----> Derive-Secret(., "c hs traffic",
                    |                     ClientHello...ServerHello)
                    |                     = client_handshake_traffic_secret
                    |
                    +-----> Derive-Secret(., "s hs traffic",
                    |                     ClientHello...ServerHello)
                    |                     = server_handshake_traffic_secret
                    |
                    +-----> Derive-Secret(., "derived", "")
```

In this context, HKDF-Extract is nothing more than a different name for the HMAC-based randomness-extraction step described in Section 5 of SP 800-56C, where the "Handshake Secret" is the (ephemeral) key-derivation key employed in the key-expansion step.

In fact, this portion of the TLS 1.3 key schedule is simply an instance of "Randomness Extraction followed by Multiple Key Expansions" as specified in Section 5.3 of SP 800-56C (assuming that a NIST-appoved HMAC is used by HKDF-Extract and assuming that (EC)DHE is formed in the context of a NIST-approved key-agreement scheme).

The other instances of HKDF-Extract in the TLS 1.3 key schedule are beyond the scope of SP 800-56C. NIST-compliance of those instances should be measured against requirements found in SP 800-108 or SP 800-133.