Public Comments on NIST Draft Special Publication 800-108

NIST received the following public comments on the draft Special Publication 800-108, "*Recommendation for Key Derivation Using Pseudorandom Functions (April 2008)*".

The comments are ordered based on the dates they were received. Most comments were received in e-mail format. The line-breaks are deleted when copied to this file. The e-mail headers are removed to protect commenter's privacy. For the same reason, e-mail addresses and telephone numbers are also removed from the signature.

Frank Ellermann

Hi, please excuse me if I miss too many points in this draft.

Apparently the four defined KDF families (counter, feedback, feedback with counter, and pipeline) can boil down to have only one iteration (c=1). If that is in fact the case it could make sense to mention corresponding length (L) values for "popular" PRFs (HMAC MD5, HMAC SHA-1, and HMAC SHA-256).

It would be also interesting to see one non-trivial example for each of the three (or four) families, with "non-trivial" meaning "at least two iterations". Some implementers might appreciate examples to see if they got it wrong. For the pipeline family even a trivial example could be interesting.

For the reasons noted in the security considerations (7.3) it can be also a good idea to note this limitation for HMAC MD5 explicitly, the consequences for longer HMAC PRFs are then more obvious.

Regards,

Frank

Jack Lloyd

Hi,

As a standard, this is specification is a disaster. Just from a quick read, I see the following:

"However, alternative orders for the input data fields may be used for a KDF."

"with a length specified by the function, an algorithm, or a protocol which uses T as an input."

"In feedback mode, the output of the PRF is computed using the result of the previous iteration and, optionally, using a counter as the iteration variable(s)."

With sufficient options, all implementations are non-interoperable. I think you've managed to reach that point here. As an implementer, my instinct is to stay well away from this entire mess and just use IEEE 1363's KDF2, which is:

- simple enough that anyone can implement it easily and without interop difficulties, or requiring protocol negotiations (and then the implementer has to do the negotiation properly – which opens up new avenues for security holes)

- secure enough that it "doesn't matter" (ie, that the likelihood that a security flaw in the KDF is the critical problem is far lower than a security flaw elsewhere in the system)

My recommendation: choose something that will work for nearly everyone, and mandate it directly. For instance, why make the counter length configurable? In 99% of implementations, the thing that will make sense is a 32-bit counter (to paraphrase the famous if apocryphal Bill Gates quote, 4 gigabytes of keying material should be enough for anybody), but by refusing to mandate this behavior, you force every implementer and application designer to choose something and then negotiate on the off chance that some other length was chosen, or that the other side is using variable length encodings - something which is allowed by the spec, as best as I can tell, and which opens up some pretty big (at least theoretical) holes.

I have no comments about the actual security aspects of it; it looks fine to my eye, but given the interoperability issues listed above I don't plan on implementing any of these KDFs anyway, so I can't say I much care whether they are actually secure or not. I would advise you to remember that crypto does not exist in a vacuum, and should help, not hinder, the overall security of a system.

Regards,
Jack Lloyd

Peter Gutmann

Jack Lloyd writes: "As a standard, this is specification is a disaster."

Somewhat more strongly worded than my comments :-), but I had the same feeling: Why yet another bunch of arbitrary PRF/KDFs to implement? We now have ones for SSL, for TLS, for SSH, for IKE, for PGP, for S/MIME, for... well I don't know every crypto protocol in existence but I'm sure there's plenty more. What's wrong with PBKDF2, which seems to do the job quite nicely? Whoever dies with the most KDFs wins?

There just doesn't seem to be any reason for this document to exist except NIH. PBKDF2 is a well-specified KDF, is relatively easy to implement (and implement in an interoperable manner), has been around for years, and has numerous interoperable implementations, including OSS ones if you don't want to implement it yourself. What's the point of SP800-108? What requirement/demand is this meeting?

Peter

Jean Campell

Classification: UNCLASSIFIED

Good morning folks,

I have briefly reviewed the subject document and noted the following comments:

a. 1. Introduction (p. 6): The first e.g. of that paragraph suggests as examples (as denoted by e.g.,) keys established using methods specified in NIST SP 800-56A and NIST SP 800-56B. Are there any other methods allowed by this recommendation? Is the AES Key Wrapping Specification (Draft) dated 16 Nov 2001 acceptable and if so, should it be specifically mentioned?

b. 1. Introduction (p. 6): In the second e.g. of that paragraph, can other EAP protocols such as EAP-FAST be included and referenced?

c. 4. Pseudorandom Function (PRF) (p. 9): In the last paragraph, suggest removing the word "Recommendations" as [1] and [2] are recommendations by themselves.

d. 5. Key Derivation Functions (KDF) (p. 10): In the second e.g. of the second paragraph, including [3] as an example of possible "mutually authenticated key establishment processes" somehow contradicts section 1 which only appears to accept [1] and [2] as the only acceptable key establishment methods, or does it? It is not clear and does not appear consistent.

Thanks for the help. This Special Publication will greatly help the Cryptographic Module Validation Program. Do not hesitate to call should you wish to discuss these comments.

Regards,

Jean

Jean Campbell, CISSP
Head - Cryptographic Module Validation Program
Industry Program Group
Communications Security Establishment Canada

Mark Manulis

Dear Authors,

I would like to comment on the additional property of collision-resistance (aka key-binding property) for the PRF functions used in your draft to derive secret keys. Collision-resistance of PRF functions is useful in case the key establishment protocol whishes to achieve the property of key confirmation. In this case it is convenient for participants to compute the output of PRF on the computed shared secret k, i.e., PRF(k, label) and exchange digital signatures on this value. In this way each protocol participant can check whether its key matches those of others by verifying the signature using own value for PRF(k, label). Collision-resistance of PRFs is thus important to provide assurance for the participants that their computed keys are identical.

You may wish to add the collision-resistance as a further property of the PRF function which should be used to derive the key.

Thank you and best regards,

Dr. Mark Manulis

UCL Crypto Group
Microelectronics Laboratory
Place du Levant 3
B-1348 Louvain-la-Neuve
Belgium

Dan Harkins

Hello,

I have read SP800-108 and have some comments, mostly around the definition of a PRF used by this specification.

A pseudo-random function is defined using a pseudo-random function family which takes the index, s, and a single input variable, x. I believe this is unnecessarily restrictive. It also appears that, because of this restrictive definition, a naive reader could implement this specification and violate a key design goal, namely that parties having a different understanding of the identities bound to the key will derive different keys.

The specification notes that there are multiple, distinct inputs to the KDF. These multiple, distinct inputs are marshaled inside the KDF and passed as a single, concatenated string to the PRF. My suggestion is to define the PRF generally to accept multiple, distinct inputs and inside the general PRF definition to define the steps one must take, if necessary, to marshal multiple inputs before being passed to the implementation-specific PRF.

In sections 5.1, 5.2, and 5.3 the inputs to the PRF include a Label and Context separated by a NULL octet. The reason for this NULL octet (noted in section 5) is as a separation indicator between "different variable length fields". (N.B.: when using the S2V PRF defined by SIV a NULL octet separator is not necessary).

The problem is that the Context, itself, can contain variable length fields-- e.g. multiple identities of the parties deriving the key-- which, themselves, may need separation. A naive reader could erroneously believe that simple concatenation of identities into a single "Context" would not require their own separator and, provided a separator is used between Label and Context, that all is well with this KDF. It is trivial to contrive two different pairs of "identities" that when concatenated together produce the same string so all would not be well with that KDF. By defining a PRF generally and describing data marshaling rules inside that general definition such a problem can be avoided.

Specific changes I suggest to address this issue are:

In section 4, I suggest defining a pseudo-random function family as "{PRF(s, x1, x2, ..., xn) | s <element-of> S} with an index s and multiple inputs x1, x2, ... xn." Each input to the PRF is a distinct string. The degenerate case is, of course, a single input: PRF(s, x).

The definition of Context in section 5 (sub 4) should state that it is: "Binary strings containing the information related to the derived keying material. It may include identities of parties who are deriving and/or using the derived keying material and, optionally, a nonce known by the parties who derive the keys. The multiple, distinct components of Context are denoted Context1, ..., ContextN".

Section 5.1 in process step 4a should be:

K(i) := PRF(KI, [i]2, Label, Context1, ..., ContextN, [L]2)

Similarly section 5.2 in process step 5a should be:

K(i) := PRF(KI, K(i-1) {, [i]2}, Label, Context1, ..., ContextN, [L]2)

And section 5.3 in process step 6b should be:

K(i) := PRF(KI, A(i) {, [i]2}, Label, Context1, ..., ContextN, [L]2)

Note that since the degenerate case of the general PRF definition is a single input no change is needed for process step 6a in section 5.3.

Then, the input data encoding in section 7.5 should describe the data marshaling steps necessary inside the generic PRF definition prior to passing the data to the implementation-specific PRF. If the underlying PRF takes a single input then all data must be marshaled into a single string. If the underlying PRF supports a vector of strings-- e.g. the S2V construct in SIV-- than no data marshalling is necessary.

What deserves emphasis in section 7.5, then, is that every variable-length input datum must be terminated before it is concatenated with another input datum. I suggest not using a single NULL octet but, instead, a 2 octet separator, encoded in "network order", of the length of the variable-length input datum.

Defining the PRF generally with data marshaling requirements inside the general definition will prevent a tragic misunderstanding of this document and will also allow other PRFs, like the S2V construct from SIV, that accept multiple, distinct inputs to treat the input data more naturally.

I also suggest some justification in section 7.4 for an iteration limit of 2^32 - 1 when a counter is not used as an input. A statement justifying the prohibition of using the KDF output as input to a stream cipher seems appropriate as well.

One last editorial comment: the second paragraph in section 7.1 has some difficult wording: "If ..., since ...." It might be clearer if it was reworded as "Since entropy of a key derivation key, generated through a key establishment protocol, is defined as a measurement of uncertainty...." I think I know what that paragraph is supposed to mean but I had to read it several times before I figured it out.

best regards,

Dan Harkins.

Thanh Nguyen

Hi,

DNI concurs with NIST Draft SP 800-108. If there are any questions or comments, please let me know.

Thanks,

Thanh Nguyen
ICTG/Policy and Planning
Office of the DNI

John Streufert

Greetings,

The Department of State concurs without comment.

Thank you.

John Streufert

IA Director

Stephen Savard

Classification: UNCLASSIFIED

Hello,

Attached are some comments from CSEC. If you put them on a website somewhere, don't attach my name to it please.

Thanks,

Steve

Stephen Savard
Cryptomath, Standards
Communications Security Establishment Canada

(Attachment: see next page)

Comments by CSEC

Clause	Location	Comment
Authority	Fifth paragraph	Change "Communications Security Establishment" to "Communications Security Establishment Canada". CSE has changed it's name to CSEC.
Abstract	Line 1	Add the word "the" before "derivation in" the first line.
Acknowledgements	Line 1 and line 3	Add the word "the" before "National".
1. Introduction	Line 2	Change "key establishment protocol" to "key establishment scheme" as stated in the abstract.
1. Introduction and 2. Scope and Purpose		Clause one has the phrase "e.g." and Clause two has the phrase "e.g.," with a comma after. Be consistent.
2. Scope and Purpose	Paragraph 1, Line 4	Change the word "process" to "scheme".
2. Scope and Purpose	Paragraph 2, Line 4 and Line 7	Change the word "agreement" to "establishment".
5. Key Derivation Functions	Paragraph 2, line 2	The term "cryptographic key" is not defined in section 3.1 Definitions.
5.3 KDF in Double-Pipeline Iteration Mode	Figure 3	Add brackets { } to the { i = 1 } { i = 2} … … { i = n }as they are optional. Figure 2 has brackets in the diagram.
7.4 Converting Keying Material to Cryptographic Keys and 7.7 Key Separation	Paragraph 1 Paragraph 1	It is stated twice that multiple keys must be from disjoint segments. It is a shall the second time in Section 7.7
Appendix A	Reference [4.] and [9.]	Reference [4.] has "pp." and reference [9.] has "pp". Make consistent.
Appendix A	Reference [10.]	This citation is missing the pp numbers.

Brian Weis

See the attached document for a few comments on this draft.

Thanks,
Brian

(attachment)

June 25, 2008
Lily Chen
Computer Security Division
Information Technology Laboratory

Dear Ms. Chen,

We am pleased to respond to the National Institute of Standards and Technology (NIST) with comments on the draft NIST Special Publication 800-108 dated April 2008. This publication describes an important and timely topic in cryptography today, namely key derivation from a secret symmetric key.

A number of new computer security protocols will be able to adopt the key derivation methods described in this publication, which will doubtless improve the security of those standards. Furthermore, as a vendor of cryptographic equipment, it is our expectation that this publication will enable computer security protocols using its methods to be more easily approved as part of the CMVP FIPS 140 program.

A few minor comments on the April 2008 draft follow.

1. We understand that *0x00* following *Label* is included in the PRF as a separation indicator, and the import of delineating the end of *Label* in application where *Label* is of a variable length in different executions. We note that applications using string with an identical length in all of its different executions may not require the separation indicator to be included in the PRF. However, we understand that requiring the null byte in all implementations results in a simpler specification.

2. This specificity of *Context* in this publication is much less than the *OtherInfo* specified in SP 800-56A, instead shifting the specificity (i.e., order and length of the *Context* components) to the application using the KDF. We support this approach, as it enables the KDF to be applied in a greater variety of applications.

3. Section 7.5 includes the following declaration: “In certain applications, additional requirements may be imposed on the encoding method.” By itself, this sentence doesn’t help the reader understand when such an application is encountered. Perhaps an example of an additional requirement will make it clearer.
We hope these comments are helpful to the NIST in developing the next version of the publication.

Please feel free to write or call with any comments or questions.

Brian Weis

Cisco Distinguished Engineer
Security Technology Group
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706

Hugo Krawczyk

Hi Lily,

I am glad to see this document coming from NIST. It is important to have a few well-defined key expansion procedures. I am also glad that you frame things, at least conceptually, in the extract-then-expand approach, even though your document is solely about expansion.

I have a few comments and suggestions.

The two main ones are:

(1) I suggest that you call the document (and the functionalities) you define in this document "key expansion" rather than "key derivation". As you explain, key expansion is only one part of the KDF process. The first one is extraction from an imperfect source but your document does not provide mechanisms for that. There is so much confusion with the term KDF, sometimes used as key expansion, sometimes as key extraction, and sometimes as both, that it would be a great contribution to the community to start cleaning up these concepts and have names that clearly differentiate one component from another. My choice is to use "extraction" and "expansion" for the different modules and "key derivation functions" for their combination. In this case all the procedures you specify are for key expansion.

(2) I suggest that you make more precise what you mean by "cryptographic key", and more specifically what exactly is a "key derivation key". This is particularly important when you say, page 10, "a key derivation key SHALL be a cryptographic key". But you do not define the latter. You only say that such a key CAN (but does not have to) be generated by a random bit generator or an authenticated key establishment process. Are there are other cases? For example, what about a pseudorandom bit generator (of the type that operating systems use to gather entropy and produce pseudo-random bits)? Also, I think that your differentiation between "cryptographic key" and "shared secret" (page 10) is rather ambiguous. It means that I cannot use g^{xy} output by a DH protocol as the key derivation key, right? But can I use half of these bits? What process needs to be applied to g^{xy} before I can use it as a key derivation key? I believe that most readers will not appreciate these subtleties without further elaboration. Certainly not those that need to implement things according to this specification. Moreover, I am not even sure that you are prohibiting the use of g^{xy} directly as a seed to the PRF, and even if you

intended to do so, I do not believe this will be clear to readers. I suggest to say this very explicitly.

In general, it would be useful to say that cryptographic keys have to be "pseudorandom strings", namely, indistinguishable from truly random strings of the same length, by any observer or attacker that is bounded to feasible computation. In particular, knowledge of some of the bits of the string provides no useful information about the remaining bits. This is not the case in general for algebraic values such as g^{xy} where, for example, some of the bits may be significantly biased from uniform.

I believe that your readers would benefit from a detailed article on the subject as the one I wrote and posted under http://www.ee.technion.ac.il/~hugo/kdf (if you are interested, at the time of the next revision of your document I can give you a more official/persistent citation for the paper than just pointing to my webpage).

Some other comments:

* Security strength. In page 8, it says that "the security strength of a key derivation function is measured by the work needed to recover either the key derivation key or the rest of the keying material when a segment of the derived keying material is known." It would be more appropriate to define it in terms of distinguishability, namely, the amount of work that an attacker needs to invest to be able to distinguish the output of the KDF from a truly uniformly distributed stream. In particular, it follows that from seeing a segment of produced keying material the attacker cannot derived any useful information on the rest of the key. (It is not only about recovering the other key bits but not even learning any information on them, e.g. the xor of the missing bits). If you adopt this type of definition you would be closer to the accepted measures of security, and also consistent with your own use (and definition) of pseudorandom functions. And if you adopt the above suggestion about defining cryptographic keys as pseudorandom strings then you will have a consistent and uniform treatment of all these notions.

* In section 4 you define the seed s to come from a set S. You may want to remark that for most PRFs (and specifically for those used here, CMAC and HMAC) the keys should be chosen uniformly (or pseudorandomly) from the set S of strings of certain size and NOT, for example, from a subset such as a DH group over strings of 1024 bits.

* I am ambivalent about the mandatory use of L as input to the PRF. I can see some advantages against the manipulation of L by an adversary. But I am afraid that this can be too restrictive in some applications. For example, an application that does not have a-priori the total number of bits that need to be derived. For example, it may need to derive 2 keys at first, but then may need to generate more key bits (this can be solved with key hierarchies, etc. but I wonder if this is not an unnecessary complication in some cases). Also, there could be cases where two parties need to access the SAME stream of bits but one needs more bits than the other (i.e., they have different L and L' but need a common prefix). It is not that I have a good specific example but I am always careful about not having over-restrictions in such general and multi-purpose standards.

* The discussion of "entropy" in section 7.1 is very sketchy. Since this is part of the security considerations for the use of the techniques in this document, this should be made more clear and accurate. For example, in the last paragraph of page 16 you say that a key establishment (KE) protocol may reduce the entropy of the key derivation key. How is a user of these specifications supposed to estimate how much entropy such a key really has (especially that strictly speaking the statistical entropy of such a key is zero!).

* 7.2 Cryptographic Strength. Taking the indistinguishability approach (namely, the amount of computation required for the attacker to distinguish the output of the KDF from purely random bits) is a better way to define these things. One has to avoid giving the impression that the only attacks of interest are those that recover the full key. See my comment above about "security strength"

* 7.3 HMAC accepts keys of different lengths but in all its analyses it is assumed that the HMAC key is random or pseudorandom. g^xy as a key is not acceptable. Please make sure to say this.

* 7.5 The requirement of 1-1 mapping is another example of too restrictive specification. For example, it would be acceptable to use the output of a collision resistant hash function instead (which is a "computational variant" of 1-1).

* I wonder what is the rationale for prohibiting the use of the output of the KDF as a key stream for a stream cipher (bottom of page 17). Can you explain?

Thanks for your work and attention,

Hugo

Nancy Cam-Winget

Hi Lily,

This is a very nice and timely written draft that will help progress the ongoing security work in the security community, in particular: 802.11 and 802.1X.

I only have two very minor comments in the attached letter.

With warm regards,

Nancy.

(See next page for attachment)

June 27, 2008
Lily Chen
Computer Security Division
Information Technology Laboratory

Hi Lily,
I am pleased to respond to the National Institute of Standards and Technology (NIST) with comments on the draft NIST Special Publication 800-108 dated April 2008. This is a timely topic in the cryptography today as many computer security protocols are requiring the use of derived keys from a shared secret symmetric key.
As a vendor of cryptographic equipment, it is our expectation that this publication will enable computer security protocols using its methods to be more easily approved as part of the CMVP FIPS 140 program.
Below are two minor comments to the April 2008 draft:

1. Section 7.5 last sentence of the first paragraph mentions "additional requirements may be imposed on the encoding method". Can a rationale and example be provided for why this sentence may be true?
2. A minor editing suggestion for the first paragraph to Section 7.7 is to break the rational for why "the keying material used by different keys need to be cryptographically separate" into bullet items for better readability.

I hope these comments are helpful to the NIST in developing the next version of the publication. Please feel free to write or call with any comments or questions.

Nancy Cam-Winget
Cisco Distinguished Engineer
Security Technology Group
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706

Sood, Kapil

Hi Lily,

Thanks for offering me the opportunity to submit some (late) comments on your KDF recommendations draft:

1. Clause 7.6: Entity Binding section
* to the maximum extent possible can be deleted, as the next sentence mentions that this binding is only possible if these have been communicated.
* The key derivation between the 2 parties may be for one of the many sessions that may be active between those 2 parties. Please include a unique Session Identifier into the KDF context string. This can be part of the entity binding clause, and your rationale in next paragraph reflects the rationale.

2. Key Name:
* Add a section that suggests defining a unique key name for every

key. Specifically, the key name should be unique and not be derived using any part of the key.

* Can the KDF be used to derive additional keying material beyond the material required by a key? Using the extra bits as a key name is acceptable (?), as long as the KDF output bit-string segments are clearly identified to be the key and key name, with no overlap. (Ref: R0-Key-Data in 11r)

Thanks for your consideration.

Best Regards,

Kapil.

Joseph Salowey

Hi Lily,

It was good to see you at the IEEE meeting in Denver. Here are some late comments on SP 800-108 (http://csrc.nist.gov/publications/drafts/800-108/Draft_SP-800-108_April-2008.pdf), I hope they are useful.

In general I think the document is good!

1. Key separation - I like that you included material on key separation as it is one of my favorite topics. Since it is one of my favorite topics and one of the main goals of key derivation I expected to see it earlier in the document. Perhaps in the definition in 3.2 and at least a mention in section 6.

A definition you could use in section 3.2 (I think the definition in 7.7 is fine, I'm just providing an alternative):

"Key separation - keys are separate if it computationally infeasible to calculate one key given knowledge of the other key."

Possible addition to section 6:

"It is often a goal to provide key separation between keys in the hierarchy such that knowledge of keys lower in the hierarchy does not make it computationally feasible to derive keys higher in the hierarchy (K1(1) and K11) and knowledge of the keys at one level in the hierarchy does not make it computationally feasible to compromise keys at the same level of the hierarchy (K11 and k12). Several parameters may be used to provide separation for the keys including purpose, consumer, and instance."

Some related thoughts on section 7.7:

I usually discuss separate in terms of purpose, consumer and instance which maps well to what you have in this section.

Purpose - label

consumers - you use identity sets which is fine. I often use consumers, because in many cases the consumer may not be the identity of a specific entity, but rather the group identifier for a set of identities.

instance - nonce or counter to separate one instance of a key from another with the same context. This is typically only needed if two keys with the same context are going to be derived.

2. Key Derivation Key Length

You reference key derivation key length in the document, however not all PRFs deal with this in the same way. For example, construction using HMAC can use a key of any length, but the construction using CMAC is limited to the block cipher length. I think it would be good to provide a mechanism for CMAC to deal with longer input than the underlying block cipher allows. For example

```
if K1 > bl then
    K1 = CMAC(0x00000000000000000000000000000000, K1)
```

Or perhaps in more general terms if K1 > pl, where pl is the maximum
length key that the underlying pseudorandom function can handle then:

```
    K1 = PRF([0], K1)
```

3. Hash function as PRF

While I am comfortable with a block cipher in CMAC mode to be a PRF I am not sure that a hash function in HMAC mode its. I think a hash function can meet the properties in section in 3.1 but could have a structured output that would make it less than suitable for a PRF. I think there may need to be more requirements on hash functions used for this purpose.

When using a hash function for a PRF I typically recommend using HMAC construction because it is well thought out and defined, however I think there can be more efficient constructions that would make suitable PRFs. Out of curiosity, did you look at these and choose HMAC because it is better or did you choose HMAC because it is common?

4. Cryptographic strength

Is the cryptographic strength, w, related to the size of the internal state of the PRF? Would this also figure into the upper bound for entropy of derived key material?

5. Entity Binding

I have mixed feelings about entity binding in key derivations. I think it is very important in certain situations and less so in others. In general I think people get a little to zealous with entity binding. Entity binding is a subset of binding to context. In general I think the section is OK except I would switch the order with key separation (generalize it to other context) or make a sub-section of key separation. Here are some additional random thoughts:

I think entity binding is primarily important if you need it provide sufficient context for key separation. That is if you are generating keys for different entities from the same root.

Key derivation failures as a means to detect protocol errors can be a bit draconian. Failure to derive keys usually means communication failure which leads to manual troubleshooting. In some cases this may be appropriate. In others there may be other consistency checks that can be performed over a secure channel that can detect problems.

I would probably rather call the section "Context Binding". Context does not necessarily need to be bound into a key derivation to be useful, it can be bound in other ways. In fact in many cases the context is often stored with the key so an entity knows when to use it. In some systems entities are issued keys instead of deriving them. The entity that binds the context to the key must communicate that context with the key.

That's all. Please, let me know if you have any questions.

Cheers,

Joe