# Public Comments on SP 800-90C (3rd Draft) Recommendation for Random Bit Generator (RBG) Constructions

Comment period: September 7 – December 7, 2022

## LIST OF COMMENTS

## 1. Comments from Marek Leśniewicz, Military Communications Institute, Poland, September 21, 2022

Dear Sirs,

I've read with interest NIST IR 8427 and NIST SP800-90C.

I believe that the study of entropy in the sense of comparing the IDEAL entropy and the REAL entropy is the only correct way in the process of assessing the randomness of binary sequences (Kolmogorov: "*information theory must precede probability theory and not be based on it*").

However, IDEAL entropy and REAL entropy have different meanings and properties that need to be analyzed and interpreted differently.

The one concerns theoretical analyzes in terms of Shannon entropy and is based on probability theory and information theory.

In this case, entropy

$$\mathrm{H}(X) = -\sum_{i=1}^{2} \mathrm{P}(X) \log_2 \mathrm{P}(X) = 1 ,$$

if we can prove that for a given sequence of random variables the probabilities P (X) = P (1) = P (0) = 1/2. It should always be remembered that in probability theory, probability is a measure of predictability, not the ratio of the number of a given event to the total number of events (Kolmogorov vs. Laplace / Buffon / von Mises).

The second one concerns practical measurements of random sequence statistics and is based precisely on mathematical statistics. In this case, we are talking about the ratio of the number of a given event $n_i$ to the total number of events $n$, measured for a given sample.

But in this case, even if the tested sample is the realization of a sequence of random variables with a proven probability P ($X$) = P (1) = P (0) = 1/2, then for any sample

$$\mathrm{H}_{\mathrm{E}}(X) \mid n) = -\sum_{i=1}^{2} (n_i / n) \log_2 (n_i / n) = 1 - \frac{2^N - 1}{2 \ln 2\, n} < 1$$

thus the entropy of the attempt $\mathrm{H}_{\mathrm{E}}(X) \mid n)$ goes to 1 with $n \to \infty$.

It is easy to check.

If we have a sample of any random sequence (also pseudo-random – DES, AES, *sponge*, SHA-3 etc.) with a size of 1 GB = 1000 MB (1 MB = 8 x 1 048 576 bits), then using the above dependence in each case we get:

$\mathrm{H}_{\mathrm{E}}(X) \mid n$ = 1 MB) = $1 - 8.6 \cdot 10^{-8}$,
$\mathrm{H}_{\mathrm{E}}(X) \mid n$ = 10 MB) = $1 - 8.6 \cdot 10^{-9}$,
$\mathrm{H}_{\mathrm{E}}(X) \mid n$ = 100 MB) = $1 - 8.6 \cdot 10^{-10}$,
$\mathrm{H}_{\mathrm{E}}(X) \mid n$ = 1 GB) = $1 - 8.6 \cdot 10^{-11}$, and for any other size $n$, respectively.

It follows that the values of H($X$) and $\mathrm{H}_{\mathrm{E}}(X) \mid n$), even for a perfectly random sequence, are never equal, i.e.

$\mathrm{H}(X) > \mathrm{H}_{\mathrm{E}}(X) \mid n)$ ,

and if the sequence is not perfectly random, then

$H(X) > H_E(X) \mid n) - e(X)$,
where $e(X)$ is a measure of the non-randomness of the sequence resulting from generator construction errors and other errors.
If this property is not included in the measurements of the string sample statistics, the comparison results of the IDEAL entropy and the REAL entropy in each case will be inconsistent.

I have described these issues in detail in my book:

Marek Leśniewicz, *Hardware generation of binary random sequences*, WAT, 2009, ISBN 978-83-61486-31-2,

unfortunately, only available in Polish (attached).

I've provided extensive summaries of the most important results of this work in two articles in English (attached):

Marek Leśniewicz, *Expected Entropy as a Measure and Criterion of Randomness of Binary Sequences* In Przeglad Elektrotechniczny, Volume 90, 1/2014, pp. 42– 46.

Marek Leśniewicz, *Analyses and Measurements of Hardware Generated Random Binary Sequences Modeled as Markov Chains* In Przeglad Elektrotechniczny, Volume 92, 11/2016, pp. 268-274.

and on the practical generation of random sequences in two others:

Mariusz Borowski, Marek Leśniewicz, *Modern usage of the "old" one-time pad*, In Communications and Information Systems Conference (MCC), 2012.

Mariusz Borowski, Marek Leśniewicz, Robert Wicik, Marcin Grzonkowski, *Generation of random keys for cryptographic systems*, Annales UMCS Informatica AI XII, 3 (2012) 75–87.

All the theoretical results presented in these papers have been repeatedly checked with measurements and they seem to be correct.

Since then, I've found several new mathematical relationships that theoretically justify some of the results obtained by the measurements. They concern entropy studies with the use of the $\chi^2$ test. I expect these papers to be published soon.

I'm eager to join the further discussion on this topic.

Best regards, Marek Leśniewicz

## 2. Comments from Jonathan P. Ng Cheng Hin, September 22, 2022

Just a small comment about the 3rd draft of 90C. I believe footnote 23 on page 39 should reference 90A's Table 3 (related to CTR_DRBG) rather than 90A's Table 2 (related to hash-based DRBGs):

- [23] For a CTR_DRBG using AES, s + 128 = the length of the key + the length of the AES block = seedlen (see Table 2 in SP 800-90A).

Thanks,

Jonathan

## 3. Comments from Ignacio Dieguez, Entrust, November 29, 2022

### NIST SP800-90C Third Draft – Entrust comments

| Type | Line# | Comment (Include rationale for comment) | Suggested change |
|---|---|---|---|
| te | n/a | There is an industry critical need to support chained DRBGs more widely in SP 800-90C,<br>e.g. being able to use RBG2/RBG3 as random sources for other RBG constructs, not just RBG1. This is the reality in many complex systems today, and will become more prevalent in the future. It can also has performance implications, if the entropy source is slow.<br><br>We acknowledge that the current draft addresses chaining for RBG1, but with the requirement/assumption that the RBG2/RBG3 used to seed the RBG1 is only used at manufacturing and is part of a separate cryptographic module, and therefore not available any more. It is understood that this is aimed at the case of resource limited crypto modules, e.g. smartcards, which may not have an entropy source available within its boundary.<br><br>This comment was already discussed with John Kelsey from NIST at the 20th Sept 2022 CMUF Entropy WG, which, based on my undertanding, he acknowledged and shared with the community that this is something that NIST supports and are currently working on defining the security requirements, which will be made available perhaps as a separate document, e.g. 90C part 2. This is very reassuring indeed and aleviates our concerns to a certain extent. I am sure that the CMUF community is willing to help NIST in defining the security requirements for chained DRBGs.<br><br>This also needs to be coordinated with the CMVP, to ensure that compliance with SP 800- 90C within FIPS 140 validations is not required before there is a way forward to certify designs with chained DRBGs. | Add chained DRBGs more widely as a valid construct in SP 800-90C.<br><br>Leverage on the input from industry experts within the CMUF to help shape adequate security requirements for these constructions. |
| te | line 953 | Terminating the RBG operation when any entropy source fails seems overly strict and does not leave room for system resiliency. As an example, let´s say a crypto module implements 3 physical entropy sources within an RBG#2 construction:<br>• 2 of them are validated<br>• 1 is non-validated, and it is used to feed the "additional input" of the DRBG<br><br>Firstly, If the non-validated source is detected to fail, it should be possible to continue normal operation, as it does not affect the claims of the certified RBG. So, the requirement in line 953 should apply only to ***validated*** entropy sources.<br><br>Secondly, it would be reasonable to allow some flexibility if only one of the validated sources is detected to fail. The DRBG, after receiving the failure signal and identifying which entropy source failed, can then pull all the required entropy from the other entropy source, which is fully functional, while the other is recovered if possible. If both entropy sources fail, then the RBG operation needs to be terminated. | A suggested change:<br><br>7. A detected failure of a validated entropy source shall cause the RBG to report the failure to the consuming application and to stop using the failed entropy source. If other validated entropy sources are available, the RBG operation may continue by making use of the healthy entropy sources only. If no other healthy entropy sources are available, the RBG operation shall be terminated. The RBG must not be returned to normal operation until the conditions that caused the failure have been corrected and tested for successful operation. |

| Type | Line# | Comment (Include rationale for comment) | Suggested change |
|---|---|---|---|
| te | 1316 | Requirement #17 seems overly restrictive, and can have performance implications in cases where the RBG2(P) has a slow entropy source. Think of a high volume production line for embedded devices with RBG1s which are being seeded by an RBG2 at the factory.<br><br>Considering that the SP 800-90B requirements and related FIPS IGs will likely make entropy sources designs much simpler, and thus likely slower, e.g. use of multiple noise sources and XOR them together is likely a design which will be difficult to justify for 90B, so developers are likely to seek simpler designs moving forward. It is also worth noting that the updated draft proposal for AIS 31 published by the German BSI in Sept 2022 have much stronger entropy requirements for the source than previous versions, i.e. 0.98 bits of min-entropy, potentially making sources that need to comply to both SP-800-90 and AIS 31 standards, slower.<br><br>From a security perspective, this requirement is forcing the RBG2(P) to operate as an RBG3(RS) construction, while there is no obvious security rationale to do so in all cases.<br><br>Instead of making this requirement mandatory ("shall"), we suggest making it recommended ("should"). | We propose that the requirement is relaxed and make it a recommendation instead of an obligation. The following is proposed:<br><br>17. If an RBG2(P) construction is used as the randomness source for the RBG1 construction, the RBG2(P) construction **should** be reseeded (i.e., prediction resistance must be obtained within the RBG2(P) construction) before generating bits for each RBG1 instantiation. |
| ed | footnote page 30 | Section 5.4.1 doesn´t exist | Update section. |
| ge | n/a | This comment is on the use of the word "pseudorandom" within this document.<br><br>Even though within the well-versed crypto community we know well the meaning of pseudo-randomness, it can have a negative impact which can, unfortunately, be exploited by the marketing departments of some vendors, e.g. to promote use of their quantum RNG products, while creating the impression of pseudo-randomness as less random, or not random at all, and not appropriate for cryptographic needs.<br><br>I would recommend that in this specification, "pseudorandom" is replaced by "random", which is already defined in the glossary (pseudorandom is not defined in the glossary) , and make the the appropriate clarifications in the glossary line 2884. | Replace<br><br>"pseudorandom"<br><br>with "random". In<br><br>the glossary,<br><br>update the<br><br>definition:<br><br>**randomness**<br><br>As used in this Recommendation, the unpredictability of a bitstring. If the randomness is produced by a non-deterministic source (e.g., an entropy source or RBG3 construction), the unpredictability is dependent on the quality of the source. If the randomness is produced by a deterministic source (e.g., a DRBG), the unpredictability is based on the capability of an adversary to break the cryptographic algorithm for producing the random bitstring. |
| Te | Line 1234 | The restrictions in lines 1234 to 1241 are not clear. Perhaps because the DRBG and sub- DRBGs are considered a single RBG1 boundary, and thus to avoid complexity of having to deal with multiple mechanisms? | Consider adding a rationale in the document to justify those requirements on RBG1 and its sub-DRBGs. |
| ed | Line 1319 | To align with ISO/IEC 19790 terminology, consider replacing "secure channel" with "trusted channel" from the current version. Note that the alternative term "trusted path" is being proposed in the 4th Working Draft.<br><br>Also, replace "physically" with "physical". | Proposed text:<br><br>A physical trusted channel **must** be used to insert the randomness input from the randomness source into the DRBG for the RBG1 construction." |

## 4. Comments from Pranshu Bajpai, Motorola Solutions, November 30, 2022

| Comment |
|---|
| Section #2.5, pg #9, Fig #2: RBG Health Test – This figure shows that a 3rd RBG health tests (top right in the figure) is required though both Entropy Source and DRBG already have such tests. What is the reason for 3rd RBG health tests? Is it optional? If so, please mention that it's optional. |
| Section 2.8, pg #12, line #702-705: "..if the status code does not indicate a success, an invalid output (e.g., a null bitstring) shall be returned with the status code if information other than the status code could be returned." – Invalid output is open ended, arguably RGB generated corrupted value can be considered as "invalid". Suggest to define the invalid output (ex, all 0s, 1s, NULL, or a fixed pattern) or mention that outputs shall be ignored in RBG returns failure. This comment also applies to other places in the document where "invalid" value is mentioned. |
| Section 2.8.1.4, pg #15-16, line 782: reseed_function input parameters. To be consistent with S800-90A, suggest to add integer prediction_resistance_request to the input list. Update the texts in this section, and Fig. #6. In many cases, prediction_resistance_request parameter is inconsistent in both SP800-90A and -90C. Suggest to make it consistent in both standards. |
| Section 3.3, pg #21, line #963-964: External conditioning function – Though SP800-90B (section 3.1.5.2) allows using non-vetted conditioning component, this [90C] standard allow only "vetted" conditioning function. To be aligned with 90B standard, suggest to allow both vetted and non-vetted condition components in 90C standard. |
| Section 4.3 pg #32, line #1247: "Sub-DRBGs can't be provide output with full entropy" – What is the reasoning for such restriction? In the next page, line #1257 it states that "note that s must be no greater than the security strength of the RBG1 construction". That means it's possible to get full entropy outputs (s bits) from Sub-DRBG when Sub-DRBG is instantiated with full entropy (s+blocklen). |
| Section 6.1, pg #44, line #1549: "If a failure is detected, the RBG operation shall be terminated" this statement contradicts requirement #8 (pg #45, line #1590-1592) which does not indicate such termination of DRBG, a failure notification is good enough. Suggest to edit. |
| General comment - 128, 192, and 256 bits security strength. Several places in 90C document stated that "at least 256 bits of entropy" (ex, line #1610) is required, but 128 bits is good enough (even after 2030 when NIST plans to retire 112 bits security strength). What is the reasoning for restricting on RGB3 (XOR) to 256 bits security strength? Can't it be flexible to 128, 192, 256 bits security strength? |
| General comment - RBG without DRBG: All 3 RBG constructions proposed in 90C draft include at least one DRBG. There might be HW modules that solely rely on SP800-90B validated TRNG. Suggest to add TRNG only option acceptable in this standard. |
| General comment: Unlike previous draft version of SP800-90C (2012 version, Figure 2, pg #13), all 3 proposed RBG construction in this new draft don't allow DRBG chaining. DRBG chaining is required in certain cases. For example, the module doesn't have any SP800-90B validated entropy source and entropy is loaded either at the factory during manufacturing or loaded into the module during provisioning through a keyloader. The module instantiates DRBG1 that saves its internal state in the non-volatile memory (ex. flash), and DRBG2 (that solely operates in volatile memory) is seeded by the DRBG1 in each power up. This new SP800-90C standard does not recognize such chaining, suggest to include this option. |

## 5. Comments from Entropy Working Group, December 6, 2022

| # | 90C location | Comment (including rationale) | Suggested change |
|---|---|---|---|
| 1 | Line 113 | In the "Note to Reviewers", point 1, you state that this draft "does not address the use of an RBG software implementation in which a) a cryptographic library or an application is loaded into a system and b) the software accesses entropy sources or RBGs already associated with the system for its required randomness."<br><br>The scenario that you do not address is a very common case; for example, the device has an HSM with a good entropy source, which we sample to seed the DRBG we use (and we can't use the HSM DRBG directly, as it is too slow for our use case). My concern is that if you finalize the draft as is, then a zealous reference lab would forbid such a common technique and insist on less secure methods for FIPS certification. And sometimes even longer chains are needed (e.g. we have an interface card without an entropy source, however we have a secure connection to the main processor which has an HSM). | We would strongly urge you to publish the eventual 800-90C with all the necessary functionality allowed. Alternately, this general design pattern could be made provisionally allowed, pending additional future rules, either through such a statement in this document, or through CMVP guidance. |
| 2 | | About RBG1s; my understanding (based on what I remember John Kelsey saying) is that it is designed so that, at construction time, the factory injects some randomness, and then when the device starts up in the field, it runs the DRBG (and updates its internal DRBG state). One practical issue that you should mandate is detecting that the stored DRBG state (which would be from the last boot time) is not corrupted. One can easily imagine a scenario where the state is corrupted (say, if the flash memory that stores the state is written too many times, or if the battery that maintains the SRAM state dies; or possibly the device loses power in the middle of the DRBG update). | There should be some mandate that requires to detect such a situation (and fail, obviously) |

| | | | |
|---|---|---|---|
| 3 | Section 4.1, 4.4.1, B.2.1 | One thing that 800-90C does not explicitly define is a ‘physically secure channel’; I wonder if you should spell it out. Here’s why: people have tried to sell me on a centralized entropy source, which distributes entropy over the network to client devices (with the entropy being encrypted as it is sent publicly). I can see them trying to claim that the client devices are RBG1 constructions, and the encrypted connection is a physically secure channel.<br><br>It should be obvious what the flaws in this system are: the encrypted packets themselves, as long as the attacker might see them, have no entropy (no matter what the centralized entropy source is); entropy is a measure of uncertainty to the attacker, and the attacker can see exactly the contents of the ciphertext packets. What the client devices will do is decrypt those packets and use that as entropy – any such entropy that the decrypted entropy will have will come from the decryption keys (and not from the entropy source itself), as those keys are the only thing the attacker might not know. | As I have run into this idea several times, it might be good to put in something which addresses this (and if a ‘physically secure channel’ is one which cannot be monitored by anyone who is not trusted, the above objection goes away). |
| 4 | Several | External conditioning. The 90C expresses cases in which external conditioning function is allowed, in which more than one entropy source contribute with their output to the conditioning function in order to obtain a full-entropy string at the output of the conditioning function.<br><br>The 90C, however, only touches (1) the case in which full entropy is desired at the output of the conditioning function, and (2) when only compliant entropy sources are providing the input to the conditioning function.<br><br>In 90B, a conditioning function is allowed to *not* provide full entropy at its output, and also is allowed to receive, as input, supplemental data, i.e., input data that contributes no entropy. The construction described in 90C does not seem to allow supplemental data (i.e., data without any claimed entropy) to be input to the external conditioning function, and also does not mention | Clarify, and allow, external conditioning functions that do receive supplemental data as part of their input (at least for conditioning functions that are not susceptible to attack by chosen supplemental data), and clarify that the external conditioning function may provide less than full entropy if the DRBG mechanism is allowed to handle those less-than-full-entropy entropy input strings. |

| | | | |
|---|---|---|---|
| | | the possibility to have external conditioning that does not provide full entropy at its output. | |
| 5 | Page ii - Question 1 : In a future revision of SP800-90C, should other constructions be included? | Why yes, yes they should. These additional external conditioning systems should be evaluated for inclusion:<br><br>1) Multi source quantum secure conditioning chains. [NS1, NS2] → Digitization → [CHT1, CHT2] →2EXT → Conditioner …<br>2) Multi source RNGs for higher throughput (entropy of all noise sources in counted) [NS+Dig]*n → conditioners →..<br>3) Multi source RNGs for reliability through redundancy. [NS + CHT]*n → [Source Allocator][cond]<br>4) Combined parallel DRBGs (E.G. non SP800-90A + SP800-90A xored together, much like SP800-90B permits non-vetted and vetted conditioners combined in a chain)<br><br>SP800-90B conditioners give a means to achieve full entropy data that in the current 90C draft can be used to seed a DRBG or feed one of the inputs of the XOR construction NRBG/RBG3.<br><br>These are proposals for major changes and so deserve justification.<br><br>1) Quantum Secure conditioning algorithms have common properties.<br>a. The proof structure is compelling – showing no external entity, even one that is fully entangled with the state of a noise source can predict the output of the deterministic PQ conditioner. This is a much stronger claim that the sort that has been made in the post quantum signature competition for instance.<br>b. They are multi source.<br>c. Being multi source, they enable conditioning algorithms based | |

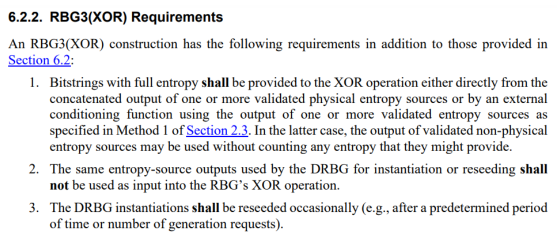
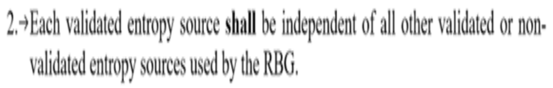
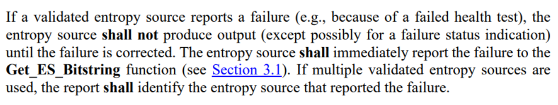
on simple linear algebra rather than complex cryptographic algorithms. No reliance on the cryptographic security of cryptographic algorithms is needed and the implementations are small, efficient and fast.

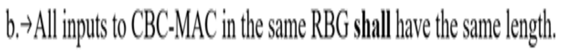
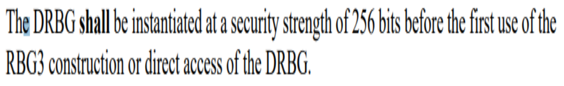
In 90B, at present quantum secure conditioners can precede a current vetted conditioner in a 90B conditioning chain, with the only disconnect being the requirement that only the entropy from a single source can be counted. To enable post quantum conditioners with simple and efficient implementation, we only need to the ability to count the entropy on each input so the addition claims of security from quantum computer based adversaries can hold.

2) Multi source RNGs for higher throughput. This is very simple. Support for fast full entropy data is something that is needed in multiple contexts. Particularly in large data centre compute situations. The computational prediction resistance claims for DRBGs falls to incrementing requirements. The O(2^128) DRBGs of recent CPUs has fallen to revised requirements for O(2^256) and this in turn is failing to meet demands for O(2^512) from some government customers.
3) Multi source entropy sources for redundant reliability. One goal of a secure system is availability. This is commonly enhanced with redundancy. The necessary self test elements are already in the standard. The standard talks with the apparent notion that there is a one-true-source while all other sources are not counted. In a redundantly reliable system, the current source can be any of the

| | | | |
|---|---|---|---|
| | | current set of not-broken sources. Explicit text making these constructs compliance should be added.<br>4) Combined parallel DRBGs. There is legitimate concern that the SP800-90A DRBGs are not secure.<br>a. The CTR-DRBG can be parameterized to create a large amount of key-reuse in AES, enabling side channel attacks.<br>b. The derivation functions do not generate full entropy data, while the 90B conditioning components do. A compliant implementation can use a DF in place of a 90B conditioner and so never be reseeded from full entropy data.<br><br>Combining DRBGs from distinct algorithms is a solution (e.g., output = XOR(CTR_DRBG_GENERATE() , CHA_CHA_Generate() ). The security falls to that of the strongest of the two algorithms. Currently, mixing the output of multiple DRBGs is common outside of NIST/FIPS contexts since there is justifiably little trust in individual DRBGs, combining them provides some resistance to discovery of attacks against an individual DRBG. | |
| 6 | **Page iii, Question: Are there any issues that still need to be addressed in SP 800-90C to allow the reuse of validated entropy sources in different RBG implementations? Note that in many** | The proposals above for more flexible use models of 90B compliant modules would enable people to take one or more ESV certified entropy sources and implement the SP800-90C constructs in software or hardware. With the tying of SP800-90C to XOR and oversampling (RS) constructions for ESV certified designs, it is not feasible to present a raw conditioner output as a primary function in a CPU instruction set. If it were and the above models were allowed, the supply of raw entropy could be as fast as a DRBG output. | |

| | | | |
|---|---|---|---|
| | **cases, specific issues need to be addressed in the FIPS 140 implementation guide rather than in this document** | The issues of handling online errors remain unclear in SP800-90A, B, C, or FIPS 140-3. This is because entropy source errors are statistical in nature and in a well-designed system will happen often as false positives. A high false positive rate is desirable since it is traded off with a low false negative error rate and so represents a conservative testing style. Yet the standard talks about errors as if they represent hard errors that lead to a failure and need to be reported. This runs counter to the 'poker face' approach of hiding detected and corrected errors in order to not hand oracles to adversaries. What would mirror the things we create for ourselves to understand the performance of our designs is to support real time health metrics that can be used as information in the policy of a cryptosystem. | |
| 7 | **Section 2.1, first paragraph** | **Current wording:**<br><br>Real-world RBGs are designed with a security goal of indistinguishability from the output of an ideal randomness source.<br><br>**Comment:**<br><br>Actually there are two goals, one for indistinguishability from uniform to a computationally bounded adversary, the other for full entropy for an NRBG, giving information theoretic security. The latter is the better goal and given sufficient performance, this is the only needed goal. DRBGs in isolation can be evaluated on the indistinguishability bound, but in SP800-90C, this is not a typical construct. As a designer of "Real-world RBGs" it's clear that non-full entropy sources are a stepping stone to full entropy only sources which will replace them all in time. | **Proposed Change:**<br><br>Both goals should be listed; therefore, the following change should be made:<br><br>"*Real-world RBGs are designed with a security goal of full entropy or indistinguishability from the output of a full entropy source.*" |

| | | | |
|---|---|---|---|
| 8 | **Section 6.2.2, item 2** | **Current wording (snapshot from draft):**<br><br>“The same entropy-source outputs used by the DRBG for instantiation or reseeding shall not be used as input into the RBG’s XOR operation.”<br><br><br>**6.2.2. RBG3(XOR) Requirements**<br>An RBG3(XOR) construction has the following requirements in addition to those provided in Section 6.2:<br>1. Bitstrings with full entropy **shall** be provided to the XOR operation either directly from the concatenated output of one or more validated physical entropy sources or by an external conditioning function using the output of one or more validated entropy sources as specified in Method 1 of Section 2.3. In the latter case, the output of validated non-physical entropy sources may be used without counting any entropy that they might provide.<br>2. The same entropy-source outputs used by the DRBG for instantiation or reseeding **shall not** be used as input into the RBG’s XOR operation.<br>3. The DRBG instantiations **shall** be reseeded occasionally (e.g., after a predetermined period of time or number of generation requests). | **Proposed Change:**<br><br>Please reword item #2 as it could be read as requiring a separate entropy source. |
| 9 | **3.2, Entropy Source Expectations, Item 2** | **Current wording (snapshot from draft):**<br><br><br>2.→Each validated entropy source **shall** be independent of all other validated or non-validated entropy sources used by the RBG.<br><br>**Comment:**<br><br>Philosophically, it seems that this requirement is impossible to wholly meet, and it is not clear how to test this requirement for positive compliance. | **Proposed Change:**<br><br>Remove requirement, or state a testable criteria for establishing what degree of mutual information is tolerable. |
| 10 | **3.2, Entropy Source Expectations, Item 6** | **Current wording (snapshot from draft):**<br><br><br>If a validated entropy source reports a failure (e.g., because of a failed health test), the entropy source **shall not** produce output (except possibly for a failure status indication) until the failure is corrected. The entropy source **shall** immediately report the failure to the **Get_ES_Bitstring** function (see Section 3.1). If multiple validated entropy sources are used, the report **shall** identify the entropy source that reported the failure.<br><br>**Comment:**<br><br>‘Immediately reporting’ a failure to a consuming application is not possible when the consuming application is not trying to interact with the entropy source. In CSP terminology a rendezvous between the sending and receiving entity is needed for the error to be | **Proposed Change:**<br><br>The language in item 6 should be refined to remove the “immediately report” language and instead indicate that the Get_ES_Biststring function will return an error if a total failure of the source is detected. There should be some acknowledgment that there may be a broader logic governing classification of a “total failure” beyond a single |

| | | | |
|---|---|---|---|
| | | communicated. So unless the receiver is ready to take the error report, the RNG cannot provide it. | health test failure (e.g., as described in SP 800-90B's "persistent error" state). |
| 11 | **3.3.1.3 Block-cipher-based Conditioning Functions, Item 2.b** | **Current wording (snapshot from draft):**<br><br>b.→All inputs to CBC-MAC in the same RBG **shall** have the same length.<br><br>**Comment:**<br><br>This appears to  impose restrictions on the CBC-MAC in order to avoid extension-style attacks without any clear technical rationale. It is not clear that extension attacks are an issue in this context. i. | **Proposed Change:**<br><br>Update to "minimum length" or justify the restriction. |
| 12 | **Figures 13 and 14 and maybe others** | **Comment:**<br><br>There appear to be wonky arrows in Figures 13 and 14. This appears to be a broad problem present in many of the diagrams. | **Proposed Change:**<br><br>Fix arrows |
| 13 | **Multiple places in sections 6.1, 6.2, and 6.3** | **Typical wording (snapshot from draft):**<br><br>The DRBG **shall** be instantiated at a security strength of 256 bits before the first use of the RBG3 construction or direct access of the DRBG.<br><br>**Comment:**<br><br>The full-entropy nature of a conditioner, an XOR construction RBG3 or an RS construction RBG3 is independent of key size or key privacy. The size of the key is immaterial to the security (notwithstanding the key weaknesses of AES256 relative to 128). If a security strength is deemed appropriate for general use (e.g., in SP 800-90A), then it should also be sufficient to | **Proposed Change:**<br><br>Remove the requirement for the DRBG component of the RBG3 to be instantiated at a security strength of 256 bits.  Any security strength allowed by 90A should be allowed here. |

| | | | |
|---|---|---|---|
| | | act as a backup in this context. The security strength of such included DRBGs should be specified on any resulting validation certificates.<br><br>This requirement would render every current Intel chip past Broadwell as non-compliant (as these designs use 128 bit keys in the conditioner and DRBG for the XOR construction RBG3). | |
| 14 | **3.2 Entropy Source requirements #7 and 7.1.2.2** | **Comment:**<br>3.2: What does “terminate the RBG operation” exactly mean? Dropping one block ? Reset ?<br><br>7.1.2.2: What does “corrected”, resp. “repaired” mean here ? | Provide further explanation in the 90C text |

## 6. Comments from Microsoft Corp., December 6, 2022

| Line | Section | Text | Comment |
|---|---|---|---|
| 113 – 116 | Note to Reviewers | "This version of SP 800-90C does not address the use of an RBG software implementation in which a) a cryptographic library or an application is loaded into a system and b) the software accesses entropy sources or RBGs already associated with the system for its required randomness. NIST intends to address this situation in the near future" | Please provide clear guidance in the final SP 800-90C and Implementation Guidance for the SP 800-90C scope to avoid confusion by module users, validators, and product developers. |
| 590 – 592 | Section 2.5<br>RBG Security Boundaries | "The RBG security boundary shall either be the same as the cryptographic module boundary or be completely contained within that boundary." | Please clarify this is the module physical boundary as described later in the SP 800-90C draft. |
| 1557 – 1560 | Section 6<br>RBG3 Constructions Based on Physical Entropy Sources | "An RBG3 construction continually accesses its entropy sources, and its DRBG may be reseeded whenever requested (e.g., to provide prediction resistance for the DRBG's output). Upon receipt of a request for random bits from a consuming application, the entropy source is accessed to obtain sufficient bits for the request." | Please consider that reseeding or getting random bits can be an optional service provided to the caller in a RBG3 construction, i.e., the DRBG implementation could initiate a reseed on its own before the seedlife such as when a timer expires. |
| 1571 – 1574 | 6.1 General Requirements | "An RBG3 construction shall be designed to provide outputs with full entropy using one or more validated independent physical entropy sources as specified for Method 1 in Section 3.3 (i.e., only the entropy provided by validated physical entropy sources shall be counted toward fulfilling entropy requests, although entropy provided by any validated non-physical entropy source may be used but not counted)." | If the entropy source was validated successfully for SP 800-90B, then using any validated should always receive credit in SP 800-90C. |

| | | | |
|---|---|---|---|
| 1712 – 1713 | 6.2.2 RBG3(XOR) Requirements | "In the latter case, the output of validated non-physical entropy sources may be used without counting any entropy that they might provide." | If the entropy source was validated successfully for SP 800-90B, then using any validated should always receive credit in SP 800-90C. |
| 1916 – 1917 | 7.2 Implementation Validation | "…and the [SP 800-90A and 800-90B] validations successfully finalized before the completion of RBG implementation validation." | Does this mean NIST is anticipating a separate certification for 90C? |

## 7. Comments from BSI, December 7, 2022

Johannes Mittmann and Werner Schindler
Bundesamt für Sicherheit in der Informationstechnik (BSI), Bonn, Germany December 7, 2022

| *Line number* | *Text passage* | *Comment* |
|---|---|---|
| 320 | Mittman | Mittmann |
| 345-346 | true random variables (variables that may be biased, i.e., each possible outcome does not need to have the same chance of occurring) | Entropy sources cannot generate random variables (mathematical construct!), but their output values can be interpreted as realizations of random variables (= values taken on by random variables).<br><br>Could the random variables not only be biased but also be correlated? |
| 457 | SP 800 90B | SP 800-90B |
| 464 | it uses dedicated hardware to provide entropy | Should this read "a dedicated hardware design"?<br>A physical noise source may be built using (carefully selected) general-purpose components. |
| 486 | j^{h} | j^{th} |
| 502-503 | However, the Method 1 or Method 2 criteria for counting entropy still applies. | It could be mentioned at this place that the noise sources should not affect each other (= part of the evaluation); see 2.6.8 (independent noise sources). Of course, otherwise, the overall entropy of an entropy string might be smaller than the sum of the entropy values of the substrings generated by the particular noise sources. |
| 668-669 | a bitstring of at least 3s/2 bits long is needed from a randomness source for an RBG1 construction, | Shouldn't it read 3s/2 bits of min-entropy? (Of course, the formulation is an immediate consequence thereof.). Requires full entropy seed string. |
| 679 | 17. The assumptions and assertions in items 3, 10, and 14 (above) apply to sub-DRBGs. | Is this enumeration correct? Item 14 refers to RBG2 and RBG3. |
| 805-818 | | Does an entropy source have to specify the length of ES_output in advance, when bits_of_entropy have been requested? Or does the calling mechanism have to support any length of output (within the range given by the 0.1 entropy per bit requirement)? |
| 875-876 | The RBG3(XOR)_Generate function (shown in Figure 10) includes a prediction_resistance_request parameter to request a reseed of the RBG3(XOR)'s DRBG instantiation, when desired. | What is the reason for a prediction_resistance_request parameter for RBG3(XOR)? The entropy source should provide sufficient fresh entropy. |

| | | |
|---|---|---|
| 963 | the vetted conditioning function listed in [SP800-90B] | Should this read “a vetted conditioning function”? (SP 800-90B lists more than one vetted conditioning functions.) |
| 1065 | 3.3 conditioned_output = Conditioning_function(input_parameters). | Is this full entropy output? |
| 1075-1076 | outlen + 64 bits | Should this read “output_len + 64 bits”? |
| 1096 | If at least n full-entropy bits have not been produced, repeat the process starting at step 3.1. | “If less than n full-entropy bits have been produced,” seems to be easier to read. |
| 1294-1295 | 8. The internal state of the RBG1 construction shall be maintained^{19} and updated to produce output on demand. | It should be ensured that output of a generate request is only output or used, after the updated internal state has been successfully stored in non-volatile memory. Otherwise, in case of e.g. a system crash, there is a risk that previously generated random bits are generated and used again. |
| 1461-1462 | This RBG may be designed to always provide prediction resistance, to only provide prediction resistance upon request, or to be unable to provide prediction resistance (i.e., to not support prediction-resistance requests during generation). | Isn't the ability to provide prediction resistance a key feature of the RBG2 construction? |

| *Line number* | *Text passage* | *Comment* |
|---|---|---|
| 1478 | (see Section 2.8.1.3 herein and in [SP800-90A]. | The closing parenthesis is missing. |
| 1504 | The RBG may include a reseed capability. | Isn't this a central feature of RBG2 constructions? (See the comment on lines 1461-1462 above.) |
| 1509-1510 | A non-validated entropy sources shall not be used for this purpose. | entropy source (singular) |
| 1547-1548 | If an entropy source fails in an undetected manner, the RBG continues to operate as an RBG2(P) construction, | Would such a DRNG be RBG2(P)-compliant? Prediction resistance cannot be guaranteed any longer even if the DRBG is reseeded (with entropy 0 in the worst case). |
| 1557-1558 | An RBG3 construction continually accesses its entropy sources, and its DRBG may be reseeded whenever requested | Should a RBG3(RS) be reseeded twice (continuously and by request)? |
| 1779 | 3.1 Obtain generated_bits from the entropy source. | Should this read “Obtain generated_bits of full-entropy DRBG output.” or similar? (The generated bits are not a direct output of the entropy source.) |
| 1806-1807 | additional_input \|\| additional_entropy | The order of additional_input and additional_entropy could be reversed, so that the Reseed step in Hash_DRBG and HMAC_DRBG is equivalent to a modified Reseed with 64 additional bits in entropy_input. |
| 1809-1810 | 256 + 64 = 384 | 256 + 64 = 320 |
| 1811-1813 | 2) concatenating the additional entropy bits with any additional_input provided in the RBG3(RS)_Generate call | In lines 1806-1807 it is the other way around, but also see the comment on those lines above. |
| 2049-2050 | These bitstrings are only unpredictable to an adversary who does not know the DRBG’s internal state. | “and who is computationally bounded” (or similar) could be added. |
| 2069-2070 | an n-bit output from the RBG3 construction is said to provide n bits of entropy. | In fact, only n(1-2^{-32}) bits of entropy are guaranteed. |
| 2154 | The personalization string to be used for this example is “Device 7056.” | The period “.” in “Device 7056.” is not part of the personalization string and should be moved outside the inverted commas. |
| 2174 | randomnessy_bitstring | randomness_bitstring |
| 2618 / 3rd line of footnote 38 | with prediction requested. | with prediction resistance requested. |
| 2656 | Both of the derivation methods specified in Appendices C.3.2 and C.3.3 an AES derivation key | A verb such as “use” seems to be missing. |

## 8. Comments from Atsec Information Security Corp., December 7, 2022

Please submit comments to: rbg_comments@nist.gov

| # | Type | Line # | Comment (Include rationale for comment) | Suggested Change |
|---|---|---|---|---|
| 1 | T | Table 1 | RBG1 and RBG3 constructions restrict the entropy source to physical. Why would the nature of the randomness affect the different RBG constructions, assuming that the entropy source is SP800-90B validated and has been certified to provide entropy? | Remove the dependency between RBG construction and the nature of the entropy (physical or non-physical). |
| 2 | T | Line 672 | FIPS 140-3 does not distinguish anymore between logical and physical boundary for a cryptographic module. Notice that IG 9.3.A, use instead "located within the physical perimeter of the operational environment" | Replace "within the physical boundary of a single [FIPS140]-validated cryptographic module", with "within the physical perimeter of a single [FIPS140]-validated cryptographic module (either the boundary of a hardware module, or the physical perimeter of the operational environment of a software or hybrid-software module). |
| 3 | | Line 1030 | The paragraph states "This construction will produce a bitstring with full entropy using one of the conditioning functions identified in Section 3.3.1.1 for an RBG2 or RBG3 construction whenever a bitstring with full entropy is required". Is the reference to section 3.3.1.1 correct, or should be 3.3.1 instead? Section 3.3.1.1 only mentions HMAC, CMAC and CBC-MAC, whereas SP800-90B referenced in line 963 mentions in section 3.1.5.1.1 unkeyed conditioning components (approved hash function per FIPS 180 or FIPS 202, Hash_df, and Block_Cipher_df) | Replace "Section 3.3.1.1" with "Section 3.3.1" |
| 4 | | Line 1196 | The length of entropy input in 1.c) should be "s+128" instead of "3s/2", as items 1.a) and 1.b). | Update item 1.c) |
| 5 | | External conditioning function | The 90C expresses cases in which external conditioning function is allowed, in which more than one entropy source contribute with their output to the conditioning function in order to obtain a full-entropy string at | Clarify, and allow:<br><br>(1) external conditioning functions that do receive supplemental data as part of their input, in which such supplemental data contributes no entropy. |

| | | | | |
|---|---|---|---|---|
| | | | the output of the conditioning function.<br><br>The 90C, however, only touches (1) the case in which full entropy is desired at the output of the conditioning function, and (2) when only compliant entropy sources are providing the input to the conditioning function.<br><br>In 90B, a conditioning function is allowed to *not* provide full entropy at its output, and also is allowed to receive, as input, supplemental data, i.e., input data that contributes no entropy. The construction described in 90C does not seem to allow supplemental data (i.e., data without any entropy) to be input to the external conditioning function, and also does not mention the possibility to have external conditioning that does not provide full entropy at its output. | (2) clarify that the external conditioning function may provide less than full entropy in its output in the cases that the DRBG mechanism connected to such external conditioning function is allowed to handle those less-than-full-entropy entropy input strings. |
| 6 | T | Lines 1579-1583 | The requirements in items 3 and 4 state that an RBG3 construction shall support, and essentially only be instantiated at 256 bits of security strength. Thus, there cannot be an RBG3 at 192 or 128 bits of security strength, regardless of whether these 192 or 128 bits are full entropy.<br><br>Because of the nature of the construction of the RBG3, and since it is being designed to provide full entropy at its output, we do not see reason to limit those RBG3 to 256 bits of strength only, if the instantiations of 128 and 192 would also provide full entropy and would be appropriate for FIPS validations at those security strengths. | Remove the limitation than an RBG3 construction supports only 256 bits of security strength by *removing item 3.*<br><br>Rewrite item 4: The DRBG shall be instantiated at its claimed security strength (128, 192, or 256 bits according to the DRBG mechanism) before the first use of the RBG3 construction or direct access of the DRBG. |