Public Comments on FIPS 202, SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions

Comment period: July 27, 2023 – October 27, 2023

On July 27, NIST's Crypto Publication Review Board initiated a review of FIPS 202, *SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions* (August 2015). The public comments that NIST received are collected below.

More information about this review is available from NIST's Crypto Publication Review Project site.

LIST OF COMMENTS

1. Comments from David O. Solé González, August 21, 2023

Hello:

I'm currently finishing a review of the document, and I found one thing that may be should be reviewed:

On page 19, specifically within algorithm 8 definition, we can read a serie of steps to do to implement said algorithm (based on current PDF found on https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.202.pdf)

1. Let P=N || pad(r, len(N)).
2. Let n=len(P)/r.
3. Let c=b - r.
4. Let P0, … , Pn-1 be the unique sequence of strings of length r such that P = P0 || … || Pn-1.
5. Let S=0^b.
6. For i from 0 to n - 1, let S=f (S ⊕ (Pi || 0 c)).
7. Let Z be the empty string.
8. Let Z=Z || Truncr(S).
9. If d≤|Z|, then return Trunc d (Z); else continue.
10. Let S=f(S), and continue with Step 8.

As far I can see, there's no definition for operation |?|, used on point 9. I can guess it's refered to the length of bit string Z, but then it can be used len(Z) right?

Kind regards,
Saludos,
David O. Solé González.

2. Comments from Dimitri John Ledkov, September 5, 2023

Dear NIST,

Blake (blake2s, blake2b, blake3) have been chosen on Linux by drivers/char/random.c, drivers/net/wireguard/, fs/btrfs, and openzfs.

This hinders interoperability with Linux kernels in FIPS mode that prohibit Blake usage. This leads to inability to connect over the network (wireguard) and read/write data (btrfs, openzfs).

Can NIST demonstrate that SHA3 (or its variants) are performant with existing Blake usage?
Or can NIST consider making Blake a FIPS approved algorithm?

It would help a lot if technologies like wireguard, btrfs, openzfs are capable of using FIPS algorithms, without compromising on performance.

Disclaimer:
I am an employee of Canonical UK Ltd (Ubuntu) but this comment is made in my personal capacity and might not represent the opinions of my employer.

--
Regards,

Dimitri.

The Test Vectors published at
https://csrc.nist.gov/Projects/Cryptographic-Algorithm-Validation-Program/Component-Testing

For "FIPS 186-4 ECDSA Signature Generation Component" currently do not include FIPS 202 SHA3 variants, could those please be included?

Also if you have any historical copies of test-vectors published there before, it would be useful to provide access to them (with stable permanent URLs).

3. Comments from Dimitri John Ledkov, October 6, 2023

Hi,

The Test Vectors published at
https://csrc.nist.gov/Projects/Cryptographic-Algorithm-Validation-Program/Component-Testing

For "FIPS 186-4 ECDSA Signature Generation Component" currently do not include FIPS 202 SHA3 variants, could those please be included?

Also if you have any historical copies of test-vectors published there before, it would be useful to provide access to them (with stable permanent URLs).

--
okurrr,

Dimitri

4. Comments from Dimitri John Ledkov, October 7, 2023

Also two versions of test vectors at
https://csrc.nist.gov/projects/cryptographic-algorithm-validation-program/digital-signatures
also do not include SHA3.

Lack of public domain informal verification SHA3 test vectors for ECDSA curves is currently hindering implementing support for ECDSA+SHA3 upstream in Linux. As I have implementation ready, but it would not be wise to add it due to lack of public domain informal test vectors. Which do exist for SHA-2 as published by NIST.

--
okurrr,

Dimitri

5. Comments from John Preuß Mattsson (Ericsson), October 24, 2023

Dear NIST,

Thanks for your continuous efforts to produce well-written open-access security documents. Please find attached our comments on FIPS 202 and SP 800-185.

Best Regards,
John Preuß Mattsson,
Expert Cryptographic Algorithms and Security Protocols

ERICSSON

Date: October 24, 2023

Ericsson AB
Group Function Technology
SE-164 80 Stockholm
SWEDEN

Comments on FIPS 202 "SHA-3 Standard" and SP 800-185 "SHA-3 Derived Functions"

Dear NIST,

Thanks for your continuous efforts to produce well-written open-access security documents. We think Keccak is a very useful primitive to build cryptographic algorithms. It is already used in e.g., TUAK [1], Ed448 [2], as well as ML-KEM, ML-DSA, and SLH-DSA [3] and it will likely be used in many future algorithms.

Please find below our comments on FIPS 202 and SP 800-185:

- *"A random function whose output length is d bits cannot provide more than d/2 bits of security against collision attacks and d bits of security against preimage and second preimage attacks"*

  We think the document should also discuss that similar considerations apply to the variable-length input message M. A random function whose input length is len(M) bits cannot provide more than len(M) security against preimage attacks. The preimage security is bounded by the Shannon entropy of the message M. If the message length is known or likely to be short, the preimage security is less than suggested in Table 4.

- We think Table 4 should be augmented with the strength against length-extension attacks. It is often claimed that security agencies from different countries participate in standardization and development of security products with the explicit goal of sabotaging security to enhance their surveillance capabilities. It is unfortunately very common that people design their own insecure "keyed hash functions" with SHA-2 by just hashing the key and the message as $\mathrm{H}(K \,||\, M)$. New examples pop up almost every year in the IETF and it likely happens very often in software projects. While we trust SHA-2 and think that it is a recommended set of hash functions, we think it is in NIST's interest to be as open as possible about the limitations of SHA-2. This would increase the trust in NIST as a global SDO for cryptography. As stated by NIST in [4], length-extension is considered a serious attack on a hash function.

- We think the specifications should describe that SHA-3 is designed to provide indifferentiability from a random oracle [5]. This is a property that modern hash functions should provide and from which pre-image, collision, and length-extension resistance follow automatically. It is also a property that many people incorrectly believe all hash functions have and a property that is required for many important uses of hash functions. The security properties listed in FIPS 202 are not enough for uses cases such as PRNGs, PRFs, asymmetric encryption padding, key derivation functions, and signature schemes that require a function producing a "random-looking" output [6]. One example is the use of SHAKE256 as a PRF in ML-KEM [7]. To use the words of John Kelsey [8], we think NIST specifications should align a bit more with the crypto community's proof-oriented worldview instead of the traditional attack-oriented worldview.

- Hardware and software APIs that only support SHA3-224, SHA3-256, SHA3-384, SHA3-512, SHAKE128, and SHAKE256 are very limiting for innovation and might significantly decrease performance (or security) of future standards. One example is ML-KEM, where the use KECCAK-p[1600, 12] would significantly increase performance. We suggest that NIST strongly recommends implementations to support KECCAK-$p[b, n_r]$. Implementations of KECCAK-$p[b, n_r]$ should be possible to test for conformance under the auspices of the Cryptographic Algorithm Validation Program (CAVP) [9], the same applies to the AES round function.

- We think NIST should add TurboSHAKE128, TurboSHAKE256, and KangarooTwelve [10–11] to FIPS 202 and SP 800-185. TurboSHAKE and KangarooTwelve are modes of operation of the KECCAK-p[1600, 12] permutation. We agree with [11] that 12 rounds give significantly increased performance without compromising security. We think TurboSHAKE is a very useful building block for MACs, encryption schemes, KEMs, and signature algorithms. A suggested outcome of the NIST workshop on encryption schemes [12] was that NIST should standardize TurboSHAKE and Rijndael with 256-bit blocks as building blocks for future encryption schemes.

- It would be good if the differences between an extendable-output function (XOF) and a variable-length hash function are explained clearly. If we understand NIST's terminology correctly, the output length of a XOF does not affect the bits that it produces, while the output length of a variable-length hash function do affect the bits that it produces.

- We think FIPS 202 should mention that the fixed-length SHA-3 functions offer meaninglessly high pre-image security significantly hurting their performance. The suggestion from NIST that "preimages need only be as hard to find as collisions" is as correct today as it was then [13]. We think the decision [14] to stick to the original requirements [4] was a mistake. The adoption of SHA-3 was hurt by the perception that it is slow, not the lack of trust [15]. As we can see, the fast SHAKE functions are much more used than the slow fixed-length SHA-3 functions. We think FIPS 202 should recommend modes of (Turbo)SHAKE and write that related outputs in XOFs can be avoided by including the length in the context. We agree with Langley that KangarooTwelve (K12) is the future. KangarooTwelve is extremely fast [16–18] with 0.51 cycles/byte on x86 and 0.75 cycles/byte on ARM. The slow fixed-length SHA-3 functions are mainly for legacy use and could be moved to an appendix.

- Mentioning of SHA-1, 160-bit digest lengths, Triple DES, and 112-bit keys should be removed from FIPS 202 as they are or will be deprecated. We think SHA3-224 should be deprecated together with Triple DES [19], which is the intended use case for SHA3-224.

- It would be good if FIPS 202 refers to SP 800-185 in the introduction and in Appendix A. An overview of all the functions with their high-level properties (fixed length, variable length, XOF, hash-function, PRF/MAC, etc.) would be nice.

Best Regards,
John Preuß Mattsson,
Expert Cryptographic Algorithms and Security Protocols

[1] 3GPP TS 35.231, "Specification of the Tuak algorithm set"
https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=2402

[2] IETF RFC 8032, "Edwards-Curve Digital Signature Algorithm (EdDSA)"
https://www.rfc-editor.org/rfc/rfc8032.html

[3] NIST, "Comments Requested on Three Draft FIPS for Post-Quantum Cryptography"
https://csrc.nist.gov/news/2023/three-draft-fips-for-post-quantum-cryptography

[4] NIST, "Announcing Request for Candidate Algorithm Nominations for a New Cryptographic Hash Algorithm (SHA–3) Family"
https://www.govinfo.gov/content/pkg/FR-2007-11-02/pdf/FR-2007-11-02.pdf

[5] Maurer, Renner, Holenstein, "Indifferentiability, Impossibility Results on Reductions, and Applications to the Random Oracle Methodology"
https://eprint.iacr.org/2003/161

[6] Green, "Indifferentiability"
https://blog.cryptographyengineering.com/2012/07/17/indifferentiability/

[7] NIST, FIPS 203 (Draft) "Module-Lattice-based Key-Encapsulation Mechanism Standard"
https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.203.ipd.pdf

[8] Kelsey, "Dual EC in X9.82 and SP 800-90"
https://csrc.nist.gov/csrc/media/projects/crypto-standards-development-process/documents/dualec_in_x982_and_sp800-90.pdf

[9] NIST, "Cryptographic Algorithm Validation Program (CAVP)"
https://csrc.nist.gov/projects/cryptographic-algorithm-validation-program

[10] Bertoni, Daemen, Hoffert, Peeters, Van Assche, Van Keer, Viguier, "TurboSHAKE"
https://eprint.iacr.org/2023/342.pdf

[11] Viguier, Wong, Van Assche, Dang, Daemen, "KangarooTwelve and TurboSHAKE"
https://datatracker.ietf.org/doc/draft-irtf-cfrg-kangarootwelve/

[12] NIST, "The Third NIST Workshop on Block Cipher Modes of Operation 2023"
https://csrc.nist.gov/Events/2023/third-workshop-on-block-cipher-modes-of-operation

[13] Kelsey, "SHA3, Where We've Been, Where We're Going"
https://csrc.nist.gov/csrc/media/projects/hash-functions/documents/burr_dimacs2013_presentation.pdf

[14] Wikipedia, "SHA-3"
https://en.wikipedia.org/wiki/SHA-3

[15] Langley, "Maybe Skip SHA-3"
https://www.imperialviolet.org/2017/05/31/skipsha3.html

[16] Team Keccak, "Is SHA-3 slow?"
https://keccak.team/2017/is_sha3_slow.html

[17] Team Keccak, "Software performance figures"
https://keccak.team/sw_performance.html

[18] Team Keccak, "KangarooTwelve: fast hashing based on Keccak-p"
https://keccak.team/kangarootwelve.html

[19] NIST, "NIST to Withdraw Special Publication 800-67 Revision 2"
https://csrc.nist.gov/News/2023/nist-to-withdraw-sp-800-67-rev-2

6. Comments from The Keccak Team, October 25, 2023

Dear NIST,

Thanks for giving the opportunity to submit comments on these documents!

We think that FIPS 202 (or SP 800-185) should define and approve the use of TurboSHAKE128 and TurboSHAKE256. On top of these, a parallelizable mode like KangarooTwelve would be a welcome addition to SP 800-185.

With their nominal 24 rounds, the XOFs standardized in FIPS 202 and SP 800-185, such as (c)SHAKE128 and (c)SHAKE256, nicely come with a very thick safety margin; e.g., the best collision attack reaches only 6 rounds. At the same time, however, an excessive safety margin hinders their full potential.

There has been sustained cryptanalysis on Keccak over the years and an amazing number of publications on this subject by the crypto community (more than on SHA-256/SHA-512). With the explicit goal of building upon this invaluable asset, we proposed in [1] the TurboSHAKE extendable output functions (XOFs) based on Keccak-p[1600] with 12 rounds. There is ample evidence that these provide a comfortable security margin, please refer to [1] for more details. Therefore, TurboSHAKE128 and TurboSHAKE256 can be seen as fast companions to their SHAKE counterparts, and we believe they provide a better combination of performance and safety margin, with the added benefit of a lower energy consumption per bit.

Another comment is about parallelizable modes. SP 800-185 provides parallelizable XOFs called ParallelHash128 and ParallelHash256. An alternative based on TurboSHAKE would yield extremely efficient XOFs on platforms that support vector instructions. An example of this is the KangarooTwelve (K12) proposal [2]. Next to the reduced number of rounds, K12 has another advantage over ParallelHash128: it does not incur a cost for short messages, thanks to the "kangaroo hopping" technique.

Kind regards,
Guido, Joan, Seth, Michaël, Gilles, Ronny and Benoît, authors of TurboSHAKE

[1] G. Bertoni, J. Daemen, S. Hoffert, M. Peeters, G. Van Assche, R. Van Keer and B. Viguier, TurboSHAKE, https://eprint.iacr.org/2023/342
[2] G. Bertoni, J. Daemen, M. Peeters, G. Van Assche, R. Van Keer and B. Viguier, KangarooTwelve: fast hashing based on Keccak-p, https://eprint.iacr.org/2016/770

7. Comments from Arne Padmos, October 25, 2023

Dear NIST team,

Please find my comments on the FIPS 202 and SP 800-185 standards and related considerations below. My main point is that NIST should consider standardising a SHAKE-based AEAD mode of operation that supports sessions, is fully committing, and retains integrity protection on nonce misuse. The process to select such a standard should be structured to ensure sufficient assurance, and should consider expected problems in real-world usage in addition to traditional cryptanalysis.

Regards,
Arne

Feedback on FIPS 202 and SP 800-185

On the general question regarding the usefulness of different algorithms in the FIPS 202 and SP 800-185 standards, I deem the SHAKE variants to be more useful than the SHA-3 variants. As to the utility of TurboSHAKE, I don't see how the benefit of a 2x increase in speed outweighs issues around compatibility with existing implementations and the increased complexity of offering yet another toggle. A more critical aspect that is missing from the SHAKE family is an AEAD mode of operation, or more specifically: one mode that is strongly committing as well as offering support for sessions while retaining integrity protection even when nonces are misused, and one SIV-based mode that retains confidentiality protection even when nonces are misused (excluding the case of equal plaintext inputs leading to equal ciphertext outputs). Of these, I deem the former to be most critical. (Note that in exploring such a mode, applicability to Ascon-p could also be considered.)

We were promised a SHAKE-based AEAD mode back in 2014, but this hasn't happened yet. On the bright side, important lessons from the previous decade can now be considered. Specifically, the ways that AES-GCM continues to fail in practice, the sorry state of protocol design, and the insights from the recent Permutation-Based Cryptography workshop [PBC23] and the NIST Workshop on Block Cipher Modes of Operation [BCM23]. Some developments to highlight include the WASI-Crypto API [Den23], flexible AE [MLH+23], and ODWrap [DMA23]. Also consider the point made in the feedback on the GCM standard that 'a recent trend to

put less responsibility on implementers of cryptographic protocols and designs seems to be gaining momentum' [NIS21] and note early work in the area of usability studies applied to cryptographic primitives [Pad23a]:
as a community, let's stop blaming the developer and let's stop giving them chainsaws when they need scissors. It is time for a SHAKE-based AEAD mode of operation with characteristics identified as useful in the recent modes workshop, e.g. support for sessions. Remember that such a permutation-based approach -- over fixing the rough edges / knife edges of e.g. GCM, which is also necessary -- would align with the trends described in the Challenges in Authenticated Encryption report from back in 2017 [ABB+17].

I'd like to emphasise that I'm not advocating for a specific algorithm. Experience with development processes has highlighted the importance of the process used for driving assurance. There are also many related questions that are not of a technical but more of a strategic policy nature. These should be formalised in a revision of NISTIR 7977 and a process taking the lessons from how standards have failed in practice should inform the development of standards going forward (see below).

Although competition dynamics are an open area of research with little work done to date -- see [Ber23] for an opinionated survey of the field and [Pad23b] for the outcomes of a related hackathon at this year's Workshop on the Economics of Information Security -- the contrast between the fates of DCM [DGW01] and OCB2 [IIMP19] highlights the importance of having development processes that offer sufficient assurance on both the design approach and sufficient adversariality in the review process (note, not in how communication happens on the forum but in whether competitors are sufficiently motivated to break other candidates). Of course, even during a competition, key implementation-related issues might not have been considered and only published years later [Ber05], or issues in leading implementations might only be found after a decade [MC23]. Also, the assurance provided by competitions in newer research areas is inherently lower, as highlighted by the attacks on F-FCSR-H [HJ08], Rainbow [Beu22], and SIKE [CD22]. Building innovative modes on top of permutations is also not without risks (e.g. see the attack on Kravatte [CFG+18]). Note that these comments apply equally to protocol design (e.g. see EES [Bla94], WEP [BHL06], SSL/TLS [MS13], TETRA [MBW23], etc.).

Even outside of competitions, there are valuable lessons to be taken from implementation problems (e.g. see the failure of early NESTOR prototypes [Boa73] and Samsung's TEE implementation [SRW22]). This

highlights potential gaps in NISTIR 7977 and CMVP, namely formalised post-market surveillance and reporting obligations on module manufacturers. Such insights can feed back into the standardisation process (such as how and when competitions are run). In additional, note that neither the informal reflection [Smi21] nor the formal evaluation [Mou21] of the AES standard asks the question of whether -- let alone how -- the cache-timing problem could have been caught during the standardisation process, and these retrospectives also don't critically reflect on the modes of operation efforts in the decades that followed the AES process. The Crypto Publication Review Project is great, but from what I can see there isn't (yet) an explicit link back to the formal standardisation process as described in NISTIR 7977.

I'd like to close with a specific example: For TETRA, SHAKE is (or was?) being considered for their new protocol suite. This effort got a helpful impetus after the TAA1, TEA1, TEA2, and TEA3 algorithms were reverse engineered and protocol-level problems were found [MBW23]. There are likely many other systems that are (re)inventing an AEAD scheme on top of SHAKE. I think it would be wise to prevent the situation we have with CBC+HMAC through standardising one or more AEAD modes of operation supporting sessions on top of SHAKE (and possibly also on top of Ascon-p).

[ABB+17] J.P. Aumasson et al. (2017). Challenges in authenticated encryption. https://chae.cr.yp.to/chae-20170301.pdf
[BCM23] The Third NIST Workshop on Block Cipher Modes of Operation 2023. https://csrc.nist.gov/Events/2023/third-workshop-on-block-cipher-modes-of-operation
[Ber05] D.J. Bernstein (2005). Cache-timing attacks on AES. https://cr.yp.to/antiforgery/cachetiming-20050414.pdf
[Ber23] D.J. Bernstein (2023). Cryptographic competitions. https://cr.yp.to/papers.html#competitions
[Beu22] W. Beullens (2022). Breaking Rainbow takes a weekend on a laptop. https://eprint.iacr.org/2022/214.pdf
[BHL06] A. Bittau et al. (2006). The final nail in WEP's coffin. https://download.aircrack-ng.org/wiki-files/doc/technique_papers/bittau-wep.pdf
[Bla94] M. Blaze (1994). Protocol failure in the Escrowed Encryption Standard. https://www.mattblaze.org/papers/eesproto.pdf
[Boa73] D.G. Boak (1973). A history of US communications security, volume 1. https://www.governmentattic.org/18docs/Hist_US_COMSEC_Boak_NSA_1973u.pdf#page=77
[CD22] W. Castryck & T. Decru (2022). An efficient key recovery attack on SIDH. https://eprint.iacr.org/2022/975.pdf
[CFG+18] C. Chaigneau et al. (2018). Key-recovery attacks on full Kravatte. https://tosc.iacr.org/index.php/ToSC/article/view/842
[Den23] F. Denis (2023). Permutation-based APIs: a framework for

future-proof cryptographic APIs. https://permutationbasedcrypto.org/2023/files/slides/PBC2023-Frank_Denis.pdf
[DGW01] P. Donescu et al. (2001). A note on NSA's DUal Counter Mode of encryption. https://people.eecs.berkeley.edu/~daw/papers/dcm-prelim.pdf
[DMA23] J. Daemen et al. (2023). Committing authenticated encryption based on SHAKE. https://eprint.iacr.org/2023/1494.pdf
[HJ08] M. Hell & T. Johansson (2008). Breaking the F-FCSR-H stream cipher in real time. https://www.iacr.org/archive/asiacrypt2008/53500563/53500563.pdf
[IIMP19] A. Inoue et al. (2019). Cryptanalysis of OCB2: attacks on authenticity and confidentiality. https://eprint.iacr.org/2019/311.pdf
[MBW23] C. Meijer et al. (2023). All cops are broadcasting: TETRA under scrutiny. https://www.usenix.org/system/files/usenixsecurity23-meijer.pdf
[MC23] N. Mouha & C. Celi (2023). A vulnerability in implementations of SHA-3, SHAKE, EdDSA, and other NIST-approved algorithms. https://eprint.iacr.org/2023/331.pdf
[MLH+23] S. Menda et al. (2023). Flexible authenticated encryption. https://csrc.nist.gov/csrc/media/Presentations/2023/flexible-authenticated-encryption/images-media/sess-7-menda-bcm-workshop-2023.pdf
[Mou21] N. Mouha (2021). NISTIR 8319: review of the Advanced Encryption Standard. https://nvlpubs.nist.gov/nistpubs/ir/2021/NIST.IR.8319.pdf
[MS13] C. Meyer & J. Schwenk (2013). Lessons learned from previous SSL/TLS attacks: a brief chronology of attacks and weaknesses. https://eprint.iacr.org/2013/049.pdf
[NIS21] NIST (2021). Public comments on SP 800-38. https://csrc.nist.gov/csrc/media/Projects/crypto-publication-review-project/documents/initial-comments/sp800-38d-initial-public-comments-2021.pdf
[Pad23a] A. Padmos (2023). Lightweight usable cryptography: a usability evaluation of the Ascon 1.2 family. https://arxiv.org/ftp/arxiv/papers/2307/2307.05504.pdf
[Pad23b] A. Padmos (2023). Making and breaking IoT protocol development and evaluation processes. https://github.com/arnepadmos/hackathon
[PBC23] Permutation-based Cryptography 2023. https://permutationbasedcrypto.org/2023/workshop
[Smi21] M.E. Smid (2021). Development of the Advanced Encryption Standard. https://nvlpubs.nist.gov/nistpubs/jres/126/jres.126.024.pdf
[SRW22] A. Shakevsky et al. (2022). Trust dies in darkness: shedding light on Samsung's TrustZone Keymaster design. https://eprint.iacr.org/2022/208.pdf

8. Comments from atsec information security corporation, October 27, 2023

Clause/Subclause(e.g. 3.1), Paragraph/Figure/Table(e.g.Table1)	Type of comment (e.g., ge = general, te = technical, ed = editorial)	Comment (Include rationale for comment)	Suggested change
Appendix A.1	ge	Appendix A.1 states that the SHA-3 hash functions and XOFs are designed to resist all attacks that a "random function" of the same output length would resist. However, the standard does not define a "random function" in the glossary. Presumably NIST refers to the common definition of a pseudo-random function, https://csrc.nist.gov/glossary/term/pseudorandom_function.	Add the definition of "pseudo-random function" to the glossary. Replace the usage of "random function" in Appendix A.1 with "pseudo-random function".
Appendix A.1	te	It would be useful if the standard includes an explicit statement that the SHA-3 hash functions and XOFs are expected to behave as pseudo-random functions. That is, the output of the functions is computationally indistinguishable from truly random output. This would provide assurance that SHAKE in particular could trivially be used as a pseudo-random number generator or key stream generator for a stream cipher.	Include an explicit statement that the output of the SHA-3 hash functions and XOFs are expected to be computationally indistinguishable from random output.
Appendix A.1	te	"moreover, if d > 1600, a preimage probably does not exist." Could this statement be clarified? Any output of SHAKE128 or SHAKE256 must have a preimage by definition. Perhaps it is meant that any random d-bit string with d > 1600 probably does not have a SHAKE128 or SHAKE256 preimage?	Clarify the statement and explain why.