Diff of NISTIR 8214A

Annotated changes between the draft version and the final version

July 8, 2020

This “Diff” highlights the changes between the draft version and the final version of NISTIR 8214A.

- The draft version, entitled “Towards NIST Standards for Threshold Schemes for Cryptographic Primitives: A Preliminary Roadmap”, was published on November 8, 2019.

- The final version changed the title to “NIST Roadmap Toward Criteria for Threshold Schemes for Cryptographic Primitives” and was published on July 7, 2020.

Some notes:

- Deletions are marked with ~~strikethrough red font~~
- Add-ons are marked with blue font
- The page numbering of this diff differs from the two base versions.
- Some changes might be not well identified by the semi-automatic markup process (using `latexdiff`). In particular, changes in the “references” section are currently not marked.
- The indices on the right-side margins map to the table in the end of this document, transcribing the received public comments, giving corresponding reply notes, and referencing to the changes induced in the document.

~~Draft~~ **NISTIR 8214A** E1: C11.1

~~Towards NIST Standards~~ NIST Roadmap Toward Criteria for Threshold Schemes for Cryptographic Primitives E2: C11.2

Luís T. A. N. Brandão
Michael Davidson
Apostol Vassilev

This publication is available free of charge from:
https://doi.org/10.6028/NIST.IR.8214A E3: C11.1

National Institute of Standards and Technology
U.S. Department of Commerce

~~Draft~~ **NISTIR 8214A**

~~Towards NIST Standards~~ NIST Roadmap Toward Criteria for Threshold Schemes for Cryptographic Primitives

Luís T. A. N. Brandão*
Michael Davidson
Apostol Vassilev
Computer Security Division
Information Technology Laboratory

** Contractor at NIST, employed by Strativia, since February 2020. Most work for this publication was done previously, as a non-contractor.* E4: C11.3

This publication is available free of charge from:
https://doi.org/10.6028/NIST.IR.8214A

~~November 2019~~ July 2020 E5: C11.1

U.S. Department of Commerce
Wilbur L. Ross, Jr., Secretary

National Institute of Standards and Technology
Walter G. Copan, NIST Director and Under Secretary of Commerce for Standards and Technology

National Institute of Standards and Technology Internal or Interagency Report 8214A
51 pages (July 2020)

This publication is available free of charge from:
https://doi.org/10.6028/NIST.IR.8214A

Certain commercial entities, equipment, or materials may be identified in this document in order to describe an experimental procedure or concept adequately. Such identification is not intended to imply recommendation or endorsement by NIST, nor is it intended to imply that the entities, materials, or equipment are necessarily the best available for the purpose.

There may be references in this publication to other publications currently under development by NIST in accordance with its assigned statutory responsibilities. The information in this publication, including concepts and methodologies, may be used by federal agencies even before the completion of such companion publications. Thus, until each publication is completed, current requirements, guidelines, and procedures, where they exist, remain operative. For planning and transition purposes, federal agencies may wish to closely follow the development of these new publications by NIST.

Organizations are encouraged to review all draft publications during public comment periods and provide feedback to NIST. Many NIST cybersecurity publications, other than the ones noted above, are available at https://csrc.nist.gov/publications.

~~**Public comment period:** ***November 11, 2019,*** **through** ***February 10, 2020***~~ **Comments on this publication may be submitted to:** E6: C11.4

National Institute of Standards and Technology
Attn: Computer Security Division, Information Technology Laboratory
100 Bureau Drive (Mail Stop 8930) Gaithersburg, MD 20899-8930
Email: nistir-8214A-comments@nist.gov

All comments are subject to release under the Freedom of Information Act (FOIA).

Reports on Computer Systems Technology

The Information Technology Laboratory (ITL) at the National Institute of Standards and Technology (NIST) promotes the U.S. economy and public welfare by providing technical leadership for the Nation's measurement and standards infrastructure. ITL develops tests, test methods, reference data, proof of concept implementations, and technical analyses to advance the development and productive use of information technology. ITL's responsibilities include the development of management, administrative, technical, and physical standards and guidelines for the cost-effective security and privacy of other than national security-related information in federal information systems.

Abstract

~~This document proposes a preliminary roadmap~~

This document constitutes a preparation toward devising criteria for the standardization of threshold schemes for cryptographic primitives by the National Institute of Standards and Technology (NIST). ~~To cover the~~ The large diversity of possible threshold schemes, as identified in the NIST Internal Report (NISTIR) 8214, ~~we tackle them in a structured way. We consider~~ is structured along two main tracks~~—~~: single-device and multi-party~~— and within each of them we consider~~. Each track covers cryptographic primitives in several possible threshold modes. The potential for real-world applications is taken as an important motivating factor for differentiating the pertinence of each possible threshold scheme. Also, the ~~standardization of threshold schemes~~ selection of items for standardization needs to consider ~~features such as configurability of parameters,~~ diverse features, such as advanced security properties, configurability of parameters, testing and validation, ~~granularity~~ modularity and composability (e.g., of gadgets vs. composites), and specification detail. Overall, the organization put forward ~~enables us to solicit feedback useful to consider~~ serves as a preparation for an upcoming solicitation of feedback useful for considering a variety of threshold schemes, while ~~at the same time considering differentiated~~ differentiating standardization paths and timelines ~~, namely depending on different~~ that may depend on the levels of technical and standardization challenges. This approach paves the way for an effective engagement with the community of stakeholders and constitutes a preparation for devising criteria for standardization and subsequent calls for contributions. While the terms standards and standardization are used throughout this report to refer to a set of possible final products, this does not imply a Federal Information Processing Standard (FIPS) as one or as the only intended format for NIST products of future threshold schemes for cryptographic primitives. E7: C11.5

Keywords~~:~~

threshold schemes; cryptographic primitives; threshold cryptography; secure multi-party computation; intrusion tolerance; resistance to side-channel attacks; standards; testing and validation; secure implementations; distributed systems.

This publication is available free of charge from: https://doi.org/10.6028/NIST.IR.8214A

This publication is available free of charge from: https://doi.org/10.6028/NIST.IR.8214A

Acknowledgments

This document follows the NIST Internal Report 8214 and the NIST Threshold Cryptography Workshop, which benefited from the participation and feedback from numerous individuals, representing various organizations. The authors of this document ~~would also like to~~ thank Nicky Mouha and Matthew Watson for ~~participation~~ participating in discussions at numerous threshold cryptography meetings, ~~and~~ Lily Chen and Andrew Regenscheid for additional feedback on ~~this draft. We welcome public feedback until this document is finalized and published .~~the initial draft, and Terry Cohen, Dustin Moody, René Peralta Isabel Wyk and the NIST Editorial Review Board for additional editorial comments. The initial draft of this report was published online on November 8, 2019, for a period of public comments. We are thankful for the valuable feedback provided by external reviewers (in chronological order): Samuel Ranellucci from Unbound Tech; Jakob Pagter from Sepior; Tore Frederiksen from Alexandra Instituttet; Svetla Nikova and Vincent Rijmen from COSIC KU Leuven; Chelsea Komlo from University of Waterloo; Nigel Smart from COSIC KU Leuven; Phillip Hallam-Baker from Venture Cryptography; Ventzi Nikov from NXP Semiconductors; Simon Hoerder and Elke De Mulder from Rambus Inc; Ronald Tse, Wai Kit Wong, Daniel Wyatt, Nickolay Olshevsky and Jeffrey Lau from Ribose Inc. E8: C11.6

This publication is available free of charge from: https://doi.org/10.6028/NIST.IR.8214A

Patent Disclosure Notice

E9: C11.7

This public review includes a call for information on essential patent claims (claims whose use would NOTICE: The Information Technology Laboratory (ITL) has requested that holders of patent claims whose use may be required for compliance with the guidance or requirements in this Information Technology Laboratory (ITL) draft publication). Such guidance and/or requirements may be directly stated in this ITL Publication or by reference to another publication. This call also includes disclosure, where known, of the existence of pending U.S. or foreign patent applications relating to this ITL draft publication and of any relevant unexpired U.S. or foreign patents .

ITL may require from the patent holder, or a party authorized to make assurances on its behalf, in written or electronic form, either: of this publication disclose such patent claims to ITL. However, holders of patents are not obligated to respond to ITL calls for patents and ITL has not undertaken a patent search in order to identify which, if any, patents may apply to this publication.

assurance in the form of a general disclaimer to the effect that such party does not hold and does not currently intend holding any essential patent claimAs of the date of publication and following call(s) ; or

assurance that a license to such essential patent claim(s) will be made available to applicants desiring to utilize the license for the purpose of complying for the identification of patent claims whose use may be required for compliance with the guidance or requirements in this ITL draft publicationeither: of this publication, no such patent claims have been identified to ITL.

under reasonable terms and conditions that are demonstrably free of any unfair discrimination; or

without compensation and under reasonable terms and conditions that are demonstrably free of any unfair discrimination.

Such assurance shall indicate that the patent holder (or third party authorized to make assurances on its behalf) will include in any documents transferring ownership of patents subject to the assurance, provisions sufficient to ensure that the commitments in the assurance are binding on the transferee, and that the transferee will similarly include appropriate provisions in the event of future transfers with the goal of binding each successor-in-interestNo representation is made or implied by ITL that licenses are not required to avoid patent infringement in the use of this publication.

The assurance shall also indicate that it is intended to be binding on successors-in-interest regardless of whether such provisions are included in the relevant transfer documents.

Such statements should be addressed to: nistir-8214A-comments@nist.gov

This call for patent claims This disclosure notice is defined in the “ITL Patent Policy — Inclusion of Patents in ITL Publications” available at https://www.nist.gov/itl/publications-0/itl-patent-policy-inclusion-patents-itl

This publication is available free of charge from: https://doi.org/10.6028/NISTIR.8214A

Executive Summary

The Computer Security Division (CSD) at the National Institute of Standards and Technology (NIST) promotes the security of implementations and operations of cryptographic primitives, such as signatures and encryption. This security depends not only on the theoretical properties of the primitives, but also on the abilities to withstand attacks on their implementations and to ensure authorized operations. To advance this capability, NIST has initiated the Threshold Cryptography project. This project intends to drive an effort to standardize threshold schemes, which enable distribution of trust placed on human operators, and offer a path to prevent several single-points of failure at the technology level.

~~The most identifiable property of threshold schemes is that they enable essential security properties — such as secrecy of~~ Threshold schemes are composed of multiple components, and assembled in a way that enables enhanced security properties and operational features even when up to f-out-of-n of their components are compromised. In such case, f is called the threshold of compromise. This enables enhanced secrecy of cryptographic keys, integrity of computed values, and/or availability of operations~~— even when up to a certain threshold number of their components are compromised. Such~~. In a dual perspective, a threshold scheme requires the correct participation of at least k-out-of-n components, for an operational goal to be achieved. For example, in a threshold Rivest–Shamir–Adleman (RSA) signing scheme with participation threshold k, the secret key is split (in a secret-shared way) across multiple signatories, such that: any subset of a threshold number k of honest signatories can produce a signature, without reconstructing the key; any subset of fewer than k signatories cannot produce a signature, nor find anything about the key. E10: C11.8

Threshold schemes can be applied to various cryptographic primitives, and ~~(for our purposes) particularly~~ in particular to NIST-approved algorithms~~, including those~~. This includes the primitives that are part of asymmetric-key schemes, such as digital signatures (in FIPS 186) and key-establishment ~~(in and)~~based on integer-factorization cryptography (IFC) (in SP 800-56B) or on discrete logarithm cryptography (DLC) (in SP 800-56A), namely elliptic-curve cryptography (ECC) [SP 800-186], and symmetric-key schemes, such as block-cipher operations (in FIPS 197). The primitives of interest encompass key generation, including requirements related to random-bit generation (in SP 800-90 series), ~~as well as the actual~~ and the related algorithms based on secret/~~private-key based algorithms~~private keys, such as signing, decryption within a public-key encryption (PKE) scheme, and enciphering and deciphering.

~~This document sets a preliminary roadmap towards~~ The structure devised in this document serves as a preparation toward the standardization by NIST, of threshold schemes for cryptographic primitives. This phase follows the publication of the NIST Internal Report "Threshold Schemes for Cryptographic Primitives" (NISTIR 8214), which positioned a preparatory framework and several representative questions, and the "NIST Threshold Cryptography Workshop" (NTCW) 2019, which brought together stakeholders to share perspectives from industry, academia and government.

The positive feedback received on the report (~~NISTIR 8214) and on the~~ NISTIR 8214) and workshop (NTCW 2019) ~~confirms~~ confirmed that there is interest and adequate knowledge by the stakeholders to initiate the process of standardization of threshold schemes. To prepare such an endeavor, this

This publication is available free of charge from: https://doi.org/10.6028/NIST.IR.8214A

document tackles the challenge of differentiating various aspects of the standardization effort, while simultaneously aiming to enable an open and transparent process with the collaboration of the community of stakeholders. This document thus defines the approaches to devise criteria for future multiple open calls for contributions for standardization, with a focus on NIST-approved primitives. This provides a number of opportunities, but also requires dealing with a number of challenges.

The main challenge is devising an effective mechanism to navigate ~~through~~ the large diversity of possible threshold schemes, ~~namely to organize, prioritize, and~~ to identify priorities and to engage with the stakeholders for collaboration and feedback. To ~~this effect~~that end, this document starts by organizing the standardization effort into two different domains: single-device and multi-party.

As confirmed by feedback in the workshop (NTCW 2019), these domains have significantly different challenges and involve different threshold considerations. ~~Within each domain we can then consider~~ Each domain can encompass various base cryptographic *primitives* and corresponding threshold *modes* of operation. ~~Each item has their specific~~ The perceived difficulty of standardization varies with the items, namely based on ~~the existence vs. absence of~~ whether or not there exist related base standards, and on the dependence on complex techniques. This makes it likely that future new standards are reached in a sequence that ~~includes first~~ first includes the simpler cases and only later the more complex cases.

Not all conceivable threshold schemes are appropriate ~~to be standardized~~for standardization. A weighting factor to consider is the potential for real-world applications, which to some extent may also affect the level of collaboration and engagement that the stakeholders are willing to undertake. An actual process of standardization also requires considering additional features, such as: the modular interplay of elements of different ~~granularity~~ complexities (e.g., building blocks vs. composites) and different levels of specification; the specification of advanced security properties (e.g., ~~about~~ composability) required for secure deployment; the suitability for testing and validation guidelines, to address regulatory requirements; and the availability of configurability options (e.g., ~~about~~ threshold values).

Using the outlined approach, this document identifies a diverse set of standardization objects (primitives and threshold modes) ~~to focus on~~on which to focus, and enumerates several features that require further consideration. The elaboration of rationale intends to serve as a basis for subsequent discussions, and help organize the collaboration with stakeholders for devising concrete criteria. Overall, the combination of the multiple aspects ~~in~~ under consideration may result in various distinct calls for contributions, as well as different timelines for the different ~~focuses. This preliminary roadmap is~~ foci. This roadmap is thus a step in a standardization process that intends to devise several useful new standards for different threshold schemes, including guidelines for testing and validation, and reference definitions of building blocks.

The end results of standardization may span new standalone documents~~as well as~~ , and be incorporated as addenda (e.g., specifying threshold modes) in existing standards, special publications, guidelines or introduced into external standards bodies. Furthermore, different items of standardization can have different associated timelines, with the latter ~~being shaped based~~ depending on the corresponding complexity of the potential threshold schemes, ~~namely with respect to~~ and on criteria

This publication is available free of charge from: https://doi.org/10.6028/NIST.IR.8214A

to be developed for their proposal, evaluation and selection.

The main purpose of this document is to ~~solicit input for our roadmap to standardize~~ prepare a rationale structure that supports an upcoming solicitation of input for useful criteria for the standardization of threshold schemes for cryptographic primitives. This process includes ~~for example~~ obtaining technical comments about threshold schemes from experts in areas of threshold cryptography, strategic comments from those who work in cryptography standards but may be unfamiliar with threshold cryptography, and input about motivating application scenarios and restrictions from security practitioners and vendors.

This publication is available free of charge from: https://doi.org/10.6028/NISTIR.8214A

Table of Contents

1 Introduction **1**
- 1.1 A multifaceted standardization effort . . . 1
- ~~1.2 A structured approach~~ . . . 3
- 1.2 A structured approach . . . 3
- 1.3 Feedback from stakeholders . . . 5
- 1.4 Organization . . . 5

2 The space of threshold schemes for potential standardization **6**
- 2.1 Two domains . . . 6
- 2.2 Primitives . . . 6
- 2.3 Modes of Input/Output interface . . . 7
- 2.4 Notions of interoperability for the client . . . 11
- 2.5 Auditability from a client viewpoint . . . 12

3 Motivating applications **14**

4 ~~Items across two tracks~~Standardization items to consider **16**
- 4.1 Multi-party track . . . 16
- 4.2 Single-device track . . . 18

5 Features of standardization items **21**
- 5.1 Validation suitability . . . 21
- 5.2 Configurability and security features . . . 21
- 5.3 Modularity . . . 23

6 Development phases **26**
- 6.1 Phase 1 — Develop ~~a preliminary roadmap~~criteria . . . 26
- 6.2 ~~Phase 2 — Develop criteria~~ . . . 26
- 6.2 Phase ~~3~~ 2 — ~~Collect and evaluate~~ Issue calls for contributions . . . 27
- 6.3 Phase 3 — Evaluation of contributions . . . 28
- 6.4 Phase 4 — Publish new standards . . . 28

7 Collaboration with stakeholders **30**
- 7.1 Multi-party setting . . . 30
- 7.2 Single-device setting . . . 30

References **32**

A ~~Application use cases~~ **34**

Appendix A — Application use-cases **34**
- A.1 Single-device encryption resistant to side-channel attacks . . . 34
- A.2 Protection of secrets at rest . . . 34
- A.3 Confidential communication . . . 35
- A.4 Decentralization of trust for key generation and distribution . . . 35
- A.5 Accountability and prevention of ill-intentioned operations . . . 36
- A.6 Distribution of trust across secure environments . . . 36
- A.7 Distributed password authentication . . . 37

This publication is available free of charge from: https://doi.org/10.6028/NIST.IR.8214A

List of Figures

Figure 1: A depiction of a variety of primitives and threshold modes across two domains 3
Figure 2: Several threshold interfaces (and one non-threshold case) 9
Figure 3: Modularity tradeoffs 24

This publication is available free of charge from: https://doi.org/10.6028/NIST.IR.8214A

1 Introduction

~~NIST~~ The Computer Security Division at the National Institute of Standards and Technology (NIST) has established the Threshold Cryptography project to drive an effort to standardize threshold schemes for cryptographic primitives. Threshold schemes enable distribution of trust placed on human operators, and ~~also~~ offer a path to prevent several single-points of failure in conventional cryptographic implementations. They often build on top of secret-sharing schemes, which split a secret into parts, called shares, such that any "share" is unintelligible on its own, but enable recovering the secret once combined into a set with a threshold number of shares. However, threshold schemes go beyond secret-sharing and enable cryptographic operations (e.g., signing, encrypting and decrypting) without ever reconstructing the key in any place. E11: C11.9

This document comes on the heels of the NIST Internal Report (NISTIR) 8214, which posed representative questions about the standardization of threshold schemes, and the NIST Threshold Cryptography Workshop (NTCW) 2019, which brought together a variety of perspectives from stakeholders.

~~The NISTIR 8214~~ The NISTIR 8214 had already identified the need to devise criteria for eventual calls for contributions for the development of new standards of threshold cryptographic schemes. ~~This~~ The present document (NISTIR 8214A) is intended to ~~devise a preliminary roadmap for the standardization effort. A main motivation~~ serve as a preparation for definition of criteria. The goal is to lay out reference rationale (complementary to what the ~~NISTIR 8214~~ NISTIR 8214 has already done), terminology, and structure that are conducive, as the project moves forward, to a precise description of the material to standardize. In doing so, the document also tries to foresee the several phases of the standardization process. This is still an early step that identifies, at a high level, the space of standardization ~~,~~ and a corresponding variety of manners to approach possible items, with possible different timelines.

~~As a roadmap tries to envision steps ahead, this document is concerned with positioning several relevant aspects towards~~ The document covers various aspects pertinent to the standardization of threshold schemes for cryptographic primitives~~. This includes~~: identifying threshold modes of interest for the primitives to thresholdize (with a focus on NIST-approved cryptographic primitives); enumerating motivating applications; specifying intended interface and security properties; devising concrete criteria for calls ~~of~~ for contribution, as well as for evaluating and selecting possible proposals, paths for testing and validation of algorithms and cryptographic modules in the threshold context; and ways of collaborating with stakeholders in an open and transparent process.

1.1 A multifaceted standardization effort

Diverse stakeholders. The challenge inherent to this standardization endeavor goes beyond the technical considerations about the simple and ~~the~~ sophisticated algorithms and techniques that enable threshold schemes for some cryptographic primitives. ~~We recognize~~ NIST recognizes a diverse set of stakeholders, including not only experts in the field of threshold cryptography, but also users, vendors, security practitioners, and those who work in cryptographic standards but may be unfamiliar with threshold techniques. The structure proposed in this document is intended to engage all stakeholders ~~and generate feedback about the roadmap~~ to generate feedback for the process ahead.

This publication is available free of charge from: https://doi.org/10.6028/NIST.IR.8214A

~~Diverse security properties.~~ Diverse security properties. The standardization of threshold schemes can promote the advancement of security related to the implementation and operation of cryptographic primitives in the real world. This is applicable to diverse security properties, such as confidentiality, integrity and availability. If systems do fail in practice ~~,~~ — often under attack ~~,~~ — due to single points of failure, then threshold schemes can enhance their protection~~, mitigating~~ . The threshold approach can mitigate the consequences of those attacks and ~~making~~ make them costlier to execute. Therefore, standardizing ~~these~~ threshold schemes may also contribute to new best security practices in cybersecurity.

~~On a variety of goals and paths.~~ On a variety of goals and paths. As the field of threshold schemes encompasses many possibilities, ~~we consider several approaches~~ several approaches can be considered across various items, not all of which fall within the scope of developing new standards. For standardization, ~~we are focused~~ the current focus is on threshold schemes for NIST-approved cryptographic primitives. ~~We want~~ The goal is to enable the standardization of threshold modes of implementation for these primitives, as a way ~~to promote better~~ of promoting an improvement of best practices in settings where the ~~use~~ implementation or operation of these primitives ~~is considered to~~ may be subject to ~~adversarial attackson the implementation or on the operation.~~ attacks.

~~There are some simple to define threshold schemes applicable~~ Some threshold schemes can be easily defined and applied to some cryptographic primitives. There are also demonstrably feasible threshold schemes whose consideration still raises difficulties for the selection of the best techniques, ~~and appropriate parameters~~appropriate parameters, and building blocks. ~~For some~~ Some of the latter ~~we still aim for standards~~are within consideration for standardization, but attaining ~~them~~ new standards will require first establishing a clear rationale to support concrete selections. Caution is needed in assessing whether particular techniques are ready for standardization, and which variations thereof are most appropriate, in particular those subject to very active research and fast-paced development. This is both a challenge and an opportunity, both of interest to a vibrant community of stakeholders. E12: C2.4

This effort will inevitably lead to some open problems of interest to the research community. For example, threshold ~~versions of~~ schemes are possible for candidate primitives under current evaluation within other NIST projects, such as the post-quantum cryptography and the lightweight cryptography~~, where the proposed conventional non-threshold primitives are still under security evaluation~~. Although interesting, these cases are not considered ~~here as in scope for standardization~~in scope here for standardization, since the proposed conventional non-threshold primitives are still under security evaluation. Nonetheless, there is interest in learning about new research results and developments in the state of the art.

~~On the types of standard/documents to produce.~~ On the types of standard/documents to produce. For some of the items identified in this document, a natural question is: ~~*do we need*~~ how useful would *a standard* be *for this* item*?* The question leaves implicit the meaning of *standard*, which may vary with the context. While the terms standards and standardization are used throughout this report to refer to a set of possible final products, this does not imply a Federal Information Processing Standard (FIPS) as one or as the only intended format for NIST products of future threshold schemes for cryptographic primitives. E13: C11.5

In some cases, a reasonable end goal may be to add ~~a simple~~ an addendum (e.g., of a simple

This publication is available free of charge from: https://doi.org/10.6028/NIST.IR.8214A

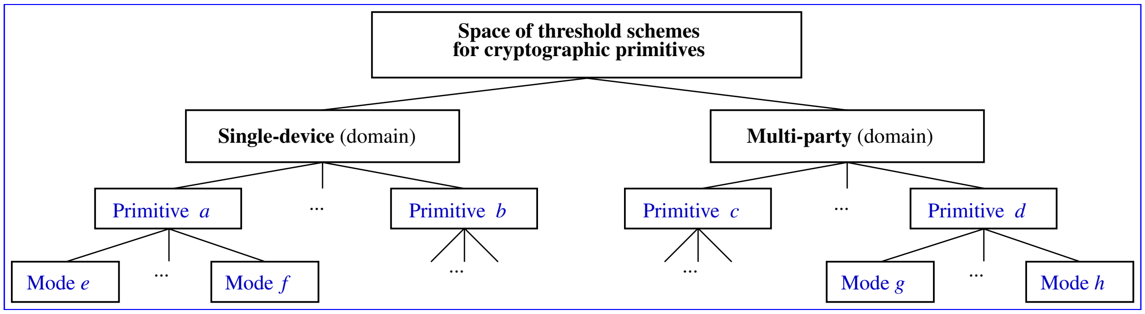


Figure 1. A depiction of a variety of primitives and threshold modes across two domains

threshold mode) in an existing standard; in others, an appropriate goal may be to devise reference definitions ~~(e.g., of secret sharing) that may appear as building block of several new techniques to consider; in some other cases~~of building blocks that may be useful for several threshold schemes to consider. In some cases, a worthy goal may be to devise implementation guidelines that enable validation within a certain security profile ~~level~~ that confirms certain threshold properties~~; in some cases we may actually consider specifying particular~~. Some cases may warrant specifying new algorithms. The concrete form in which to deliver the new standards will become apparent as ~~we move~~ the process moves forward.

A key ~~takeaway~~note**:** ~~we want to engage with stakeholders towards~~ Engaging with stakeholders is a priority in this project, toward an informed definition of criteria for standardization of threshold schemes for cryptographic primitives.

1.2 A structured approach

1.2.1 ~~The potential space of standardization~~

1.2.1 The potential space of standardization ~~Since~~

Considering several dimensions of the space of threshold schemes~~has many dimensions~~, the analysis of potential items for standardization benefits from a structured approach. ~~We start by distinguishing~~ It is useful to first distinguish between the single-device and the multi-party *domains* ~~. In~~ (Figure 1). Then, in each domain there ~~is a potential applicability for~~ are several cryptographic *primitives* ~~, and each of those can be potentially implemented~~ that can be considered, each with a potential for implementation in various *modes*. However, not every conceivable possibility is suitable for standardization. Simplicity of standardization does not necessarily imply that an item should be standardized. Similarly, a perceived difficulty ~~need-not keep us away from advancing towards~~ need not prevent advancement toward standardizing an item, even if it may take longer to achieve.

1.2.2 Motivating applications

While there are many conceivable threshold schemes, ~~we consider~~ for this project it is important to focus on where there is a high need and high potential for adoption. An overarching motivation in this effort is developing the ability to distribute trust in operations, and ~~increasing~~ increase resistance against attacks on implementations~~, of~~. This applies to NIST-approved cryptographic primitives, since they already underpin the security of many real systems. Several potential applications can benefit directly from the threshold properties enabled in implementations of these cryptographic

This publication is available free of charge from: https://doi.org/10.6028/NISTIR.8214A

primitives. ~~We can benefit in learning from stakeholders about more concrete applications~~ Along the process, stakeholders can provide feedback on concrete applications of interest.

1.2.3 Items across two tracks

~~As a main organization level, we consider two separate standardization~~ At a high level, the standardization effort is organized into two separate **tracks** ~~— one per domain (~~(one per domain): single-device and multi-party~~)~~. The two domains differ substantially in system ~~model~~models, so the separation ~~in tracks allows us to better differentiate various~~ into tracks allows a better differentiation between concurrent approaches of standardization.

~~For each track we are interested in organizing~~ The organization of possible items (primitive/mode) for standardization can be independently organized per track. Some of the default ~~potential primitives to~~ primitives to potentially consider for thresholdization come from NIST standards specifying the Rivest–Shamir–Adleman (RSA) signature and encryption schemes, the Elliptic Curve Digital Signature Algorithm (ECDSA), the Edwards Curve Digital Signature Algorithm (EdDSA), the Elliptic Curve Cryptography (ECC) Cofactor Diffie-Hellman (CDH) primitive, the Advanced Encryption E14: C7.2
Standard (AES), and methods for random number generation (RNG). ~~Within these, there~~ There is a special interest in the primitives related to secret keys, such as key-generation, signing, decryption within a public-key encryption (PKE) scheme, and symmetric-key enciphering and deciphering. For each ~~primitive we are interested in considering what are the~~ of them, it is important to identify relevant threshold modes of operation, and how some of their technical challenges may vary with respect to standardization.

1.2.4 Detailed features

Besides the ~~high level~~ high-level identification of threshold modes of interest, there are detailed features of fundamental importance in the upcoming phase of criteria definition. ~~This preliminary roadmap emphasizes three aspects :~~ Three important aspects are:

- *configurability and security features*~~— need to be specified in order to~~ , whose specification is needed to characterize the threshold scheme, including its interface;
- *suitability for validation*~~—~~, required in the process of allowing the use of cryptographic schemes in several application scenarios (e.g., in the U.S. federal context); and
- *modularity of components and specification detail*~~— relevant to identify~~ , which is relevant to identifying recurring building blocks (~~such as~~ e.g., secret sharing) that may appear across several threshold schemes, as well as improving the security analysis and the simplicity of specification.

1.2.5 Development phases

~~We intend to drive the standardization project in~~ The standardization project encompasses phases of devising criteria for calls for contributions, evaluating proposed contributions, and writing documentation for new standards. Standardization items with different ~~development~~ developmental needs may be organized into different tailored calls for contributions and corresponding timelines. This improves

This publication is available free of charge from: https://doi.org/10.6028/NIST.IR.8214A

collaboration with a set of stakeholders interested in a variety of standardization items and challenges. Expected new standards and guidelines may include reference definitions (e.g., for secret sharing), algorithms/techniques for threshold implementations, and security profiles for validation/certification. The resulting documentation may span a variety of formats, including addenda to existing standards (e.g., a simple threshold mode of operation), and new standalone documents (e.g., describing new complex techniques and analysis).

1.3 Feedback from stakeholders

To drive an open and transparent standardization process, ~~the~~ several phases present opportunities for public feedback. Currently, ~~we are particularly interested~~ there is particular interest in the following topics:

1. standardization items (~~inc.~~ including threshold modes) ~~fitting~~ that fit the described organization;
2. potential real-world applications ~~motivating~~ that motivate concrete threshold schemes;
3. interface and security properties of interest in the threshold scope;
4. criteria for evaluating and comparing ~~between~~ a variety of possible instantiations; and
5. forms of collaboration with stakeholders.

1.4 Organization

Section 2 outlines a mapping of the potential standardization space ~~,~~ into specification levels of domains, primitives, and threshold modes. Section 3 considers application motivations for threshold schemes. Section 4 discusses concrete primitives and threshold modes of interest in the multi-party and ~~in the~~ single-device domains. Section 5 emphasizes several features whose consideration is required when specifying criteria for concrete items. Section 6 discusses the generic phases of development ~~towards~~ toward new standards. Section 7 proposes and motivates high-level aspects of criteria and calls for contributions from stakeholders. Appendix A describes examples of motivating applications.

This publication is available free of charge from: https://doi.org/10.6028/NISTIR.8214A

2 The space of threshold schemes for potential standardization

2.1 Two domains

To organize As a way of organizing the potential space of standardization of threshold schemes, we start by distinguishing this project distinguishes two domains: single-device and multi-party. The denominations intend to be literal: the former refers to a single device that internally confines all logical components of the threshold scheme; the latter refers to a threshold system composed of multiple parties, possibly with independent locations. The **single-device** domain is associated with a rigidity of configuration of components, strictly defined physical boundaries, and a dedicated communication network. Conversely, the **multi-party** domain intends to enable tends to enable a modularized patching of components (e.g., repairing newly found bugs in existing components, or even entirely replacing old components by with new ones) and may allow dynamic configurations of the parties in a protocol (possibly decided by an administrative authority). The multi-party case may also require solving problems related to distributed systems, such as byzantine agreement (consensus).

The two domains share common features with respect to certain threshold elements, and some aspects may be cross-domain applicable. For example, secret-sharing as a technique is often a basic component applicable to both domains. Furthermore, the two domains can also be applied hierarchically, such as in a multi-party threshold implementation where each party is itself a thresholdized single-device.

2.2 Primitives

In the scope of this standardization endeavor, the [cryptographic] *primitive* layer is a main aspect of characterization of an item for thresholdization. We distinguish several primitives The same conventional scheme (e.g., "encryption scheme"), often defined as a tuple of algorithms, can encompass several primitives of interest (e.g., key-generation vs. encryption vs. decryption)that are often associated within the same conventional scheme (e. g., "encryption scheme"). This separation . The separate consideration of primitives allows modularizing distinct concerns of single-points of failure, which may be considered differently across application settings. For example: on one hand, the ability to avoid a *dealer* of a secret key (i.e., having a dealerless scheme) may be a desirable feature for some application scenarios, but we do not see a dealer as an inherent shortcomingof a ; on the other hand the use of a dealer is not necessarily a shortcoming, e.g., in certain application settings where the dealer is the enabler of a subsequent threshold scheme. Therefore, the need for threshold key-generation should be considered separately from the need for threshold signing, decryption or enciphering. In Section 4 we focus Section 4 focuses on some NIST-approved algorithms defined in Federal Information Processing Standards (FIPS) and Special Publications in Computer Security (SP 800). Overall, these include concrete instantiations for : signing, decryption (within a public-key encryption (PKE) scheme), enciphering/deciphering, and key generation (including RNG), and one-side key-operations related to pair-wise key-agreement. E15: C7.2

The process of developing new standards must include establishing a clear rationale to support concrete selections. Therefore, it is likely that the first new published standards will stem from simple techniques capable of thresholdizing already current NIST-approved algorithms. One probable

This publication is available free of charge from: https://doi.org/10.6028/NIST.IR.8214A

example, simple and concrete, is that of a threshold version of RSA signing or decryption, where the private RSA key is initially secret-shared across several parties. This can be instantiated in a way that n-out-of-n or even k-out-of-n ~~manner~~parties need to be present to produce the signature or decryption. When a cryptographic operation is required, each party individually computes something with their secret share, and later the outputs are combined, without ever combining together the shares that would enable recovering the secret key. Other simple examples can include threshold schemes resulting from simple combinations of techniques similar or closely related to those standardized, as may happen to achieve some multi-signatures with independent keys.

~~Even the above simple example already illustrates how a technique enables distributing~~ Several threshold techniques enable distributing — across several parties or components — the trust about the secrecy of a private key~~. Then,~~ , such that the compromise of the internal state of a single party ~~does~~ would not completely break the security of the system. When having to sign or decrypt a plaintext, the set of parties operates in such a way that the end result is as if a cryptographic module held the key at some point in time, but in fact the result is obtained without the key ever being recombined in a particular place.

With respect to publishing standards, ~~over time we will reach cases~~ it is likely that with time it will be possible to reach schemes that require more complex compositional design approaches, possibly using some building blocks that do not currently appear in any NIST standard. ~~This is nonetheless focused on schemes with~~ The overall design of those schemes must have well-understood security properties~~of the overall design~~. Since the base primitives of focus are NIST-approved cryptographic primitives, the task of analyzing the security and parameters of the original non-threshold algorithm is likely to not be ~~an hindrance for~~ a hindrance to the standardization process. For example, threshold RSA key generation can be comparatively difficult, but the decision of which parameters to use for RSA keys is already dealt with at the level of the non-threshold primitive. Rather, in such cases, the complexity of standardization is in specifying the building blocks, defining a protocol for a chosen threshold mode (~~see~~ Section 2.3), and analyzing the security of the composition.

2.3 Modes of Input/Output interface

Before thresholdization, the conventional paradigm of interest is one where a client requests an operation from a cryptographic *module*, as depicted in Figure 2a. The client first sends, to the module, a *request* with some input, ~~e.g.,~~ such as a plaintext p for encryption or for signing, or a ciphertext c for decryption); then the client receives back the *reply* with the intended output, ~~e.g.,~~ such as a ciphertext block $c = \text{AES}_K(p)$, ~~or~~ a signature $\sigma = \text{ECDSA}_K(p)$, or a decrypted plaintext $p = \text{RSA}_K(c)$, where K denotes the secret/private key.

At a high level, ~~we consider~~ a similar paradigm can be considered for threshold schemes, with respect to a *client*, with some input, requesting that some entity processes a cryptographic primitive. However, as a fundamental difference, the entity receiving and processing the request and outputting its result is a *threshold entity*, which is~~in fact~~, in fact, a composite of components (either multiple parties, or a single-device with several components) enabling a threshold property for some security property. ~~In the perspective of the client, the threshold entity can still be abstracted as a cryptographic module (and in some cases may even be indistinguishable from a conventional one), although possibly with some additional sophistication in the interface and/or on how to interpret the input and output.~~

This publication is available free of charge from: https://doi.org/10.6028/NIST.IR.8214A

~~We define the *threshold mode* as a~~ Within the client–module paradigm, this document uses the notion of threshold mode to refer to the level of characterization ~~used to distinguish properties of the threshold scheme in the~~ that distinguishes, from the perspective of the client, variations of the threshold scheme. It is thus possible to refer to multiple distinct thresholdized modes for the same cryptographic primitive. Note: the meaning of "mode" here should not be confused with the usage in "block-cipher mode of operation", which identifies how a block-cipher can be used to encrypt and decrypt large messages. E16: C11.10

~~Figure 2 also depicts several distinct interfaces for the threshold case: no I~~From the perspective of a client, the threshold entity can still be abstracted as a cryptographic module (and in some cases may even be indistinguishable from a conventional one). Nonetheless, this may possibly involve some sophistication in the interface and/~~O secret-sharing (Figure 2b), secret-sharing of both~~ or on how to interpret the input and output~~(Figure 2c), secret-sharing of only the input (Figure 2d), secret-sharing of only the output (Figure 2e).The figures are mere abstractions.The actual communication medium and the input/output connections depend on the implementation and on a more detailed specification of the threshold scheme~~. While it is relevant to keep it simple on the client-side, some sophistication is allowed: the use of client-side secret-sharing (or reconstruction), or the ability to perform additional verifications (e.g., signature verifications). E17: C10.4

~~The following are two possible aspects of characterization of a threshold mode:~~

~~**input/output interface (on the client)** — whether or not the client needs to perform secret sharing of the input and~~With respect to the input/~~or secret reconstruction of~~ output (I/O) interface, Figure 2 depicts four distinct cases: no I/O secret-sharing (Figure 2b), secret-sharing of both input and output (Figure 2c), secret-sharing of only the input (Figure 2d), and secret-sharing of only the output (Figure 2e). The figures are mere abstractions. The actual communication medium and the ~~output; and~~

~~**auditability** — whether or not the client can prove that an obtained output was produced by a threshold scheme (e. g., identifying *k* components with registered identities in some public-key infrastructure).~~

~~Other threshold mode aspects may be considered along the standardization process.~~ input/output connections depend on the implementation and on a more detailed specification of the threshold scheme.

2.3.1 ~~Input/output interface~~

~~With respect to the input/output (I/O) interface, we distinguish four cases:~~

The following description conveys additional detail about the interfaces:

- **Not-shared-IO:** ~~the client sends~~ The client sends the full input to the threshold entity (via a relaying proxy or primary component, or by broadcasting to all components)~~the full input~~, and later receives back the output, exactly as in the non-threshold scheme. See Figure 2b.
- **Shared-I:** ~~the~~ The client secret-shares the input in a k-out-of-n manner~~;~~, and then sends each share to each component of the threshold scheme~~; the~~. The components may then communicate between themselves to securely compute the output (e.g., a ciphertext c) without learning the input. This mode is relevant for enhanced secrecy of the input ~~,~~(e.g., a plaintext submitted for symmetric encryption, or possibly even for signing~~.~~). See Figure 2d.
- **Shared-O:** ~~upon~~ Upon the completion of a threshold computation, each component obtains only a secret share of the output (e.g., of a decrypted plaintext), and sends it to the client~~; the~~. The

This publication is available free of charge from: https://doi.org/10.6028/NIST.IR.8214A

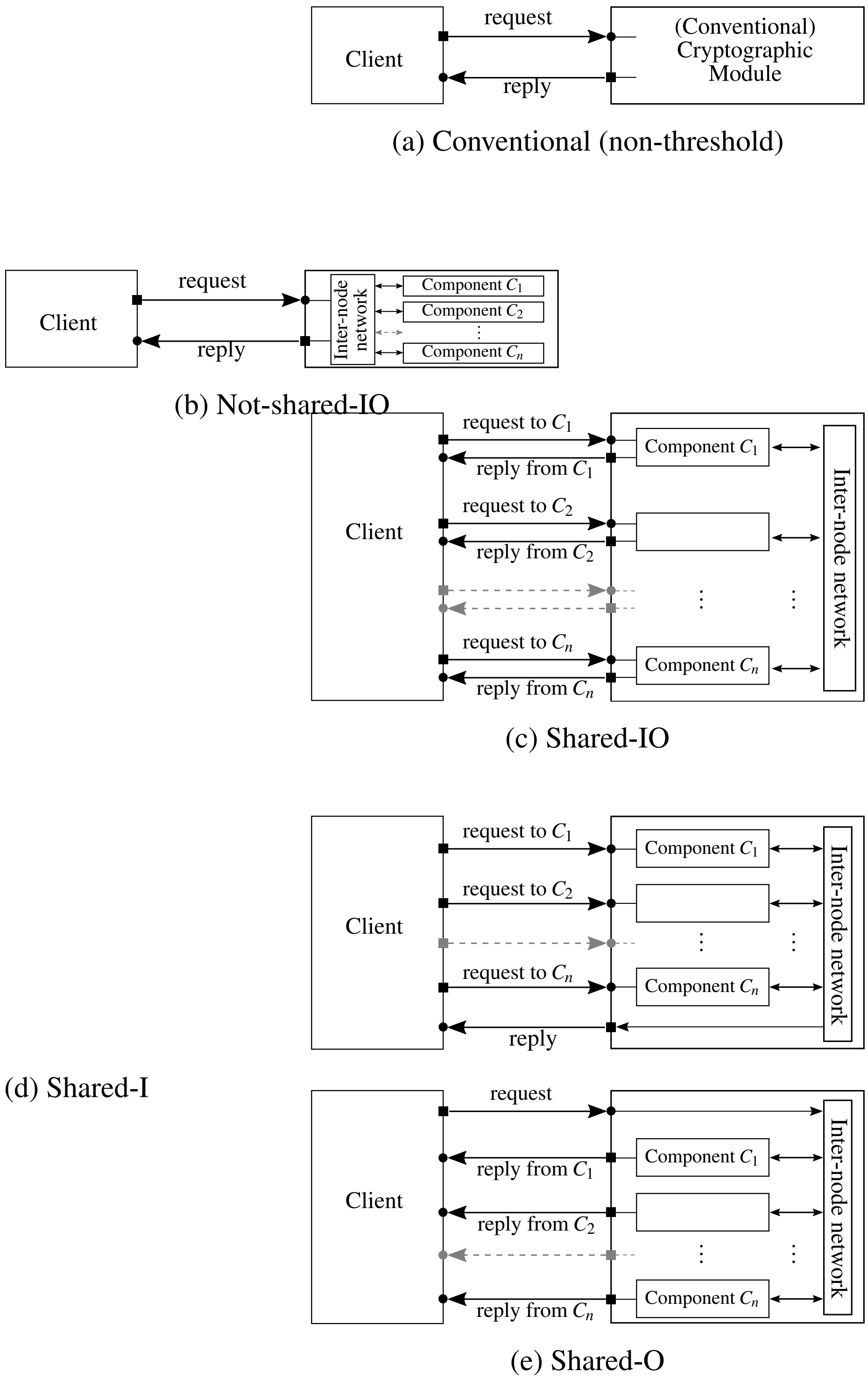


Figure 2. Several threshold interfaces (and one non-threshold case)

client then reconstructs the final output from the shares. This mode is relevant for enhanced

This publication is available free of charge from: https://doi.org/10.6028/NIST.IR.8214A

secrecy of output , (e.g., a plaintext obtained from threshold decryption.). See Figure 2e.

- **Shared-IO:** both Both the input (I) and the output (O) are secret-shared across the components of the threshold scheme. Only the client sees the complete input and output. See Figure 2c.

Note: we use "shared-I"Shared-I/O" is used to denote any case within shared-I, shared-O, and shared-IO. Other threshold mode aspects may be considered throughout the standardization process.

Note on key generation. Note on key generation. The above distinctions apply well to primitives with a clearly defined inputand output , namely those primitives /output (e.g., ciphertext/plaintext). This includes cases where the needed secret or private key has already been secret-shared in advance. The case of key generation as a primitive can be slightly different, if the administrator client does not intend to learn the generated secret (symmetric) or private (asymmetric) key , but rather intends the threshold entity (module) to be updated with a new internal secret-shared key. In that case, the client uses as input input sent by the client to the threshold entity is a key length and some generic protocol parameters, different from an actual input for signing or encryption/decryption. As output of the threshold computation, the client receives a public-key, if applicable, and nothing else (apart from protocol metadata, e.g., a confirmation of success). Nonetheless, the shared-I/O mode is still conceivable, if useful for some application. For example, the client could provide some of its the input (e.g., a base element of a public key) in a shared-I mode, and/or the "public key" could be calculated in a shared-O manner, such that the client would collect those shares and locally calculate the public keylocally. .

Note on intermediaries. Note on intermediaries. A not-shared-IO mode may in some cases be achieved based on a shared-I/O mode, by incorporating in the threshold entity an intermediate secret-sharing / reconstructor proxy mediating the communication between the client and the threshold components (except if the underlying shared-I/O mode requires communication authentication between the client and components). In a not-shared-IO mode, the client may or may not be aware of the threshold nature of the cryptographic "module".

Note on other schemes. While some of the shared-INote on the scope of possible modes. With respect to standardization, it is important to consider the dimension of interoperability between threshold and non-threshold schemes (Section 2.4), particularly from the perspective of a client requesting a service (e.g., signature, encryption, decryption) from a cryptographic module. The possible adaptation of clients to perform secret-sharing and/O modesaddress privacy concerns about the inputor output, there are more sophisticated schemes where not even a full collusion of the components/parties of the threshold scheme would learn anything from the input. Those schemes, or reconstruction is an exception that yields the various shared-I, shared-O and shared-IO modes. Some additional aspects of client involvement may be implicit within communication protocols, such as possibly using transport layer security (TLS) between the client and the threshold entity. E18: C2.6

The presented interface alternatives, limited to secret-sharing, do not cover all conceivable multi-party protocols, but they address some privacy and integrity concerns about input/output. Some conceivable alternatives requiring sophisticated client enhancements are left out of the current scope. For example, one could conceive interactions where the client does not would interact in a

This publication is available free of charge from: https://doi.org/10.6028/NIST.IR.8214A

secure two-party computation with the module (threshold entity) in order to let go of the secrecy of the input and output, even if the module is not thresholdized~~, are possible for example based on secure two-party computation. These schemes fall outside~~ . This would be a case where even a collusion of all of the components/parties of the threshold scheme would not learn anything about the input of the client. This approach falls outside of the direct scope of the threshold cryptography project, but ~~are~~ is within the area of interest of the privacy-enhancing cryptography project.

2.3.1 ~~Auditability~~

~~We denote a mode as *auditable* if~~

2.4 Notions of interoperability for the client

It is useful to characterize the interoperability of a threshold scheme from the perspective of a client. Section 2.3 discussed the input/output interface nuances. The present section considers the functional properties of the output of a cryptographic primitive. In particular, if a client is capable of interpreting and operating on outputs of the non-threshold scheme, then it is relevant to determine whether the client is also able to handle the final output produced by the threshold implementation. The functional scope refers to the ~~client is able to verify and prove to a third party that the obtained result was generated from a threshold execution. This property is for example obvious in a signature defined as a concatenation of signatures, since the client can later show several signed components. Perhaps less obvious, but quite useful, is the case of concisemulti-signatures whose size is independent of the number of signing parties, and whose verification is similar to that of the~~ main output, ignoring whether it may have had to be reconstructed from shares (when in a shared-O/IO mode), and whether or not there may exist additional protocol metadata made available to the client (e.g., public-keys and/or authentication by each threshold component). For the purpose of this document it is useful to define some terminology. E19: C11.10

Functional equivalence. A threshold scheme for a cryptographic primitive is functionally equivalent to its non-threshold ~~signature. These schemes define a procedure whereby the client determines an 'equivalent' public-key corresponding to the combination~~counterpart if for each input in the threshold scheme the output probability distribution is identical. Examples of functional equivalence: E20: C11.11

- Block-cipher. Since a block-cipher is a function, a threshold implementation of it must lead to the same output as the non-threshold counterpart (possibly after computations related to the input/~~aggregation of keys of the involved parties, such that a successful signature verification based on the derived public key implies that the several parties have participated.~~ output interface mode).

  ~~Auditability~~
- Decryption. A decryption algorithm, upon inputted with a well-formed ciphertext, must return the corresponding unique plaintext that was encrypted. The equivalence applies for decryption with any secret key possible in the threshold scheme, even if the key generation itself is non-equivalent (example in next bullet), as long as it is secure.

Functional interchangeability. A threshold scheme for a cryptographic primitive (e.g., signing) is functionally interchangeable, with respect to a subsequent operation (e.g., verification), if the final output of the threshold executing of the primitive can be used instead of the output of the conventional E21: C11.12

This publication is available free of charge from: https://doi.org/10.6028/NIST.IR.8214A

(non-threshold) primitive, with respect to the referred operation that uses said output as an input. Examples of functional interchangeability:

- Key-generation is interchangeable with respect to signing and decryption, if the generated keys are compatible with those operations. For example, a distributed RSA key generation may be functionally interchangeable even if it biases the probabilistic distribution of the output keys to be RSA moduli with both primes equal to 3 mod 4. E22: C3.6
- Probabilistic vs. deterministic signatures can be interchangeable if the client can verify them using the same algorithm, for each possible public key. For example, a probabilistic signature using secret randomness may, with respect to signature verifiability, be functionally interchangeable with a version that uses pseudo-randomness seeded with the secret key, but it is not equivalent since the distribution is altered. This difference is non-trivial for some applications, as it may, for example, be relevant for applications relying on determinism for recreating the signature. Still, when focused on verifiability a client with access to the verification key can verify signatures regardless of whether they came from the non-threshold scheme or the interchangeable threshold one.

Functional interchangeability is being considered with respect to honest executions, rather than as an assertion about security. However, for standardization the interest is to consider schemes that are secure with respect to defined properties (e.g., unforgeability) and/or ideal functionalities. Thus, a threshold scheme that is functionally interchangeable but not equivalent to some non-threshold primitive needs to have its security properties more carefully assessed in the new context. For example, while conditioning the primes of an RSA modulus to be 3 mod 4 may pose no security threat, there would be a problem if the primes were limited to a small set.

It is important to consider interoperability when evaluating threshold schemes for standardization. Functional interchangeability is one useful notion, among other possibilities. Other scopes of interoperability may be considered orthogonal to the aspect of I/O interface. For example, a as well. For example, deployment-wise it is relevant to consider how much of a client implementation can remain the same.

2.5 Auditability from a client viewpoint

For the purposes of this document, a threshold scheme is called functionally auditable if the client can prove to a third party that the output of the cryptographic primitive was generated in a threshold manner. Examples:

- Additional data. Besides the information needed to reconstruct the intended output, the client may receive from each participant component of the threshold scheme a (PKI verifiable) signature of the output.
- Multi-signatures. The public-keys of each participating signatory are combined into a single public-key for signature verification (e.g., functionally interchangeable with EdDSA), such that the public-key combination also enables later proving which independent signers participated in the signature. (This requires the infeasibility of finding two distinct sets of public keys that yield the same combined public key.)

Orthogonality to the I/O interface. Although a shared-I/O mode may be auditable, the mode itself

This publication is available free of charge from: https://doi.org/10.6028/NIST.IR.8214A

does not imply auditability(even though the . Even though a client uses secret-sharing), since the final reconstructed output may be equal to one from a conventional implementation, without a way to externally prove , the client may remain unable to externally prove that a threshold computation . A took place. Conversely, a not-shared-IO mode may allow auditabilityin the case where there is , if the client receives complementary information (e.g., zero-knowledge proofs, or transcripts of authenticated communication with multiple components) allowing authenticated transcripts) that allow an external verification of the participation of multiple components with registered identities.

2.5.1 Interchangeability

Compatibility with functional interchangeability. Depending on the details, it is possible that "auditability" and "functional interchangeability" are simultaneously achievable. For example, this happens if (i) there is audit-enabling metadata (which the client can decide to use or not), and (ii) the client can still extract from the received information an element that is functionally interchangeable with the corresponding non-threshold output. E24: C3.2

We call a mode *interchangeable* if the input and output communication of the client is as in the conventional implementation primitive. This implies in particular the use of a not-shared-IO mode. It is worth noticing that there may be not-shared-IO modes that are not interchangeable. This happens for exampleif the output (not secret-shared) is authenticated by all participating parties (e.g., via signatures vouching for the correct output), which the client needs to parse to decide on the correctness of the output, but which are themselves not part of the final output.

Auditability may be a desirable feature for some use-cases, but it is not a necessary requirement. For example, a threshold deployment scenario may favor privacy of the identities of the threshold components, which in turn may be achievable to the detriment of auditability.

This publication is available free of charge from: https://doi.org/10.6028/NIST.IR.8214A

3 Motivating applications

The selection of items (primitive–mode) of interest for standardization should consider potential applications taking advantage of threshold schemes for cryptographic primitives. ~~This~~ The consideration of applications can help foresee potential deployment scenarios and be useful to tailor future calls for contributions. It can also help characterize the set of stakeholders potentially interested in providing contributions to the standardization effort. Motivation may come from:

- **Deployed applications** ~~,~~ making use of threshold schemes, despite lack of standards (or NIST standards)~~— the~~. The development of new standards can promote best practices and interoperability in a field with already concretely demonstrated use-cases.
- **Potential applications** ~~,~~ whose deployment would be facilitated by new standards for threshold schemes. Particularly, for widely used NIST-approved cryptographic (key-based) primitives ~~, we consider that~~ a default motivation for thresholdization is the ability to distribute trust across several operators.

A strong motivation for achieving threshold properties in a cryptosystem implementation is to reduce its susceptibility to single points of failure. These failures can often affect a combination of confidentiality, integrity, and availability. Correspondingly, threshold schemes can be designed to enhance a combination of properties, often with tradeoffs. Usually, some form of secret sharing or distributed key generation is employed in order to initially distribute trust, across multiple parties or components, on the protection of a secret. Other threshold schemes can then retain this distribution of trust while the shared key is used to perform cryptographic operations.

In the multi-party domain, the distribution of shares across multiple parties can enable removing single points of failure of ~~availability~~ various kinds. For example: of availability, by not requiring all parties to be present~~, of confidentiality~~; of confidentiality, by requiring a greater number of colluding parties to find the key~~,~~; and of integrity, by implementing robust techniques that detect and address faults from malicious parties.

In the single-device domain, the goal is also to prevent key-leakage, e.g., from exploitation by side-channel and fault-injection attacks, and can include improving integrity and availability. A threshold circuit design can prevent the secret key from being in an identifiable location, thereby making its leakage much more difficult. For example, certain exploits may then require collecting a number of traces that is exponential in the number of secret shares.

For the multi-party domain, ~~we focus~~ the focus is on applications in the active model, where corrupted parties can deviate arbitrarily from the protocol specification. As such, ~~we~~ it is relevant to consider enabling verification of correctness of a produced output (or contributed share). For the single-device domain there is also interest in exploring schemes with active security, but ~~we also see~~ there is also value in developing passively secure schemes against key-leakage.

Appendix A describes potential application use-cases, such as ~~:~~ single-device encryption resistant to side-channel attacks~~;~~, protection of secrets at rest~~;~~, trust decentralization for key generation and distribution~~;~~, accountability and prevention of ill-intentioned operations~~; confidential communication;~~

password authentication; , confidential communication, password authentication, and interacting hardware security modules.

This publication is available free of charge from: https://doi.org/10.6028/NIST.IR.8214A

This publication is available free of charge from: https://doi.org/10.6028/NISTIR.8214A

4 Items across two tracksStandardization items to consider

The development of standards for threshold schemes has a potential to improve best-practices in the implementation and operation of cryptographic primitives. However, the matter of deciding which items to standardize, which techniques to support them, and which new documents to emerge as standards, is a complex matter that requires careful ponderation and a participative process. In this process it is useful to identify commonalities and synergies that may ease the development of standards for a variety of items. E25: C11.13

This section describes at a high level some technical aspects required for threshold schemes for some primitivesand subject to and modes being pondered for standardization. Since the two domains (multi-party and single-device) correspond to substantially different implementation scenarios, we also refer to their corresponding processes as different standardization their corresponding standardization processes are referred to as different *tracks*. Furthermore, also within each domain, we briefly describe issues that may there are issues that potentially differentiate items in terms of being considered *simple* vs. *more complex*, which in turn hints at different standardization timelines and paths.

We put a stronger initial emphasis The initial emphasis is on obtaining threshold versions of NIST-approved conventional primitives. Some threshold schemes are simple, originating from well defined techniquesmay originate from simple well-defined techniques, already based on properties of the underlying cryptographic primitive. Other cases may require more complex techniques, e.g., generation, use and verification of correlated-randomness in the single-device domain, and building blocks from secure multiparty computation in the multi-party domain.

Note. Some trivial threshold schemes are left out of the scope of the following discussion. For example, we ignore threshold schemesSection 4.1 and Section 4.2 leave out of scope some trivial threshold schemes, such as those based solely on trivial concatenation (e.g., of signatures), or nesting (e.g., of encryption, in a cascade mode), or of repetition from multiple implementations of approved conventional primitives implemented with independent keys. Conversely, a related but within scope case is that of multi-signatures, which, despite being usable in a setting with multiple independent (public/private) keys pairs, enable producing concise signatures with size independent of the number of participants.

We do not assume the following lists The set of items identified ahead is not assumed to be exhaustive.

4.1 Multi-party track

4.1.1 Simpler cases

RSA signing.

4.1.1.1 RSA signing. The essential challenge for producing a threshold RSA signature is in thresholdizing the modular exponentiation, which needs the secret key and the hashed-and-encoded plaintext as input. The hashing-and-encoding can be performed by the client, or by a proxy, or (if it is not a problem to leak the clear plaintext) by the components of the threshold entity. We focus The focus is on obtaining a not-shared-IO mode. The There is also interest in considering

This publication is available free of charge from: https://doi.org/10.6028/NIST.IR.8214A

the shared-I mode~~may also be of interest, case in which the hash-and-encode is performed by the client , to avoid threshold hashing.~~ , in which case the client would secret-share the hashed-and-encoded the plaintext. (These two modes are considered of interest for all upcoming mentioned signing primitives.)

~~**RSA decryption.** We consider the mode, which is essentially the same as considered for signatures~~

4.1.1.2 RSA decryption.

The primitive is similar to RSA signing, except that the input is a ciphertext and the output is a (possibly encoded) plaintext. There is a default interest in the not-shared-IO mode. Since the plaintext is the usual object of confidentiality ~~concerns~~concern, for the decryption operation ~~we also envision as potentially relevant~~ it may also be relevant to consider the shared-O mode, i.e., as an enhanced way of preventing leakage of sensitive data. E26: C11.10 E27: C11.14

~~**EdDSA signing.**~~

4.1.1.3 EdDSA signing.

The EdDSA[1] is a deterministic variant of the Schnorr signature. There are probabilistic Schnorr signatures that can be easily thresholdized, in a simultaneously auditable and ~~mode~~functionally interchangeable manner, with the signature being similar in syntax to an original non-threshold signature, and with the verification key depending on the set of participating signers for each signature~~, but the signature still being similar in syntax to an original non-threshold signature~~. The concrete (deterministic) EdDSA replaces the randomness by a hash of the concatenation of the secret signing key and the message being signed. ~~This~~ Thus, a threshold version of this EdDSA signature requires distributed hashing, where the hash function needs to be a NIST-approved function within the Secure Hash Algorithm (SHA) families SHA2 or SHA3. (Note: In the "HashEdDSA" version, the pre-computed hash of the message, instead of the possibly longer message, is used as input to the distributed hashing in a threshold implementation.) This is significantly costly to do in a distributed way for SHA2 or SHA3. This creates a technical difficulty (substantial inefficiency) for achieving a corresponding ~~threshold mode, which~~ functionally equivalent threshold mode, compared to what would be possible for a probabilistic signature. This may either imply ~~for it~~ a more complex ~~longer~~ path of standardization, or additional possible considerations about the exact intended threshold mode. E28: C6.1

~~**Key generation for elliptic curve cryptography (ECC).**~~

4.1.1.4 Key generation for elliptic curve cryptography (ECC).

For EdDSA and ECDSA signatures, the secret key is a multiplicative factor (in elliptic curve notation) that ~~leads a public generator into~~ operates on the public generator to produce the public key. ~~The generation of secret~~ Secret keys for the mentioned elliptic-curve signatures can be easily ~~performed~~ generated from independent random shares. To ensure that each party ends with an actual random share, the distributed key generation may also include multiparty coin-flipping and commitments to the shares held by every party. The consideration of threshold techniques for key generation for ECC should take into account the NIST recommendations (e.g., SP 800-186) on parameters acceptable for elliptic curves. E29: C7.23

[1] Considerations about EdDSA are based on the FIPS 186-5 draft, which may still be adapted in its final version.

4.1.1.5 Pair-Wise Key-Establishment Schemes Using ECC. Pair-wise key agreement is a fundamental tool for many two-party protocols. There are simple techniques to achieve it, such as those described in SP 800-56A based on discrete-log cryptography. These can be based on operations over elliptic curves, which may be easy to thresholdize due to inherent homomorphic group operations. Even though key-agreement is a protocol with more than one party, its thresholdization is considered per party, with respect to the operations with secret keys. For example, in an ECC Cofactor Diffie-Hellman (ECC-CDH) key agreement, the basic building block is simply a "multiplication" with a secret key (the equivalent of an "exponentiation" if done under integer finite-fields). With respect to homomorphic properties this is somewhat similar to the RSA threshold case, but with the simpler setting of a known group order. E30: C7.2

4.1.2 More complex cases

~~RSA key-generation.~~

4.1.2.1 RSA key-generation. Threshold modes of interest for RSA key-generation require multiple parties jointly computing a public modulus without any threshold set learning anything secret about the prime factors, along with all of the parties learning secret shares of the secret decryption/signing key d. This can be achieved based on secure multi-party computation, and there are implementations that demonstrate its feasibility.

~~ECDSA signature.~~

4.1.2.2 ECDSA signature. A technical difficulty in threshold ECDSA is ~~in~~ jointly computing a secret sharing of a multiplicative inverse of an ~~additively secret shared~~ additively-shared secret value. This is less straightforward than a simple homomorphic computation (e.g., as in the case of threshold RSA), but ~~can nonetheless~~ it can, nonetheless, be feasibly performed based on state-of-the-art techniques. ~~We are interested~~ There is interest in the not-shared-IO mode, possibly simultaneously auditable. Being a signature, the shared-I mode may also be of interest.

~~AES enciphering and deciphering.~~

4.1.2.3 AES enciphering and deciphering. The mathematical structure of the AES S-Box (the non-linear component of AES) does not provide homomorphic properties enabling an easy thresholdization in the multi-party setting. Nonetheless, threshold versions can be implemented based on techniques of secure multiparty computation. Threshold versions of enciphering and deciphering can be of interest in the shared-I and shared-O modes, respectively. Both primitives can also be relevant in ~~an~~ a not-shared-IO mode.

4.2 Single-device track

Historically, cryptographic algorithms were implemented in hardware devices long before cryptography appeared in software. As software cryptographic implementations started to dominate the

This publication is available free of charge from: https://doi.org/10.6028/NISTIR.8214A

This publication is available free of charge from: https://doi.org/10.6028/NIST.IR.8214A

mainstream technology used at home and the office, people again turned to hardware for acceleration and security. For example, AES instructions and Secure Hash Algorithm (SHA) SHA extensions were provided on Intel x86, AMD and ARM processors. More recently, as the complexity of single-chip devices increased and the emergence of Systems on a Chip (SoC) technology became mainstream, more complete implementations of cryptographic capabilities appeared in hardware. For example, the there has been a rapid and accelerating growth of Field Programmable Gate Arrays (FPGA) devices in recent years in response to existing and emerging computational needs in different domains, including deep learning and artificial intelligence, bring opportunities in using . Consequently, the FPGA platform can be used as both an accelerator for cryptographic algorithms and as a host platform with cryptographic capabilities intended to protect the intellectual property of the customization logic programmed on the platform.

One of the most widely implemented algorithms in hardware is AES. At the same time, it is well-known that hardware implementations of cryptographic algorithms, AES in particular, bring specific security challenges to the table. Side channel leakage has been a difficult problem for hardware manufacturers over the years. In practice, the hardware industry relies on empirical and expensive techniques to mitigate the potential leakage weakness of cryptographic algorithm hardware implementations. There is a significant industry need for implementing AES in a way that provides a better mitigation of side-channel leakage in hardware.

4.2.1 Simpler cases

AES enciphering with masked input.

4.2.1.1 AES enciphering with masked input. Leakage resilience can be achieved based on masking techniques for generic Boolean circuits. This involves a secret-sharing of the input key material so that each wire or register only "sees" a share, and never an actual secret bit. Furthermore, the protection needs to be propagated across the circuit path, in order to prevent leakage of sensitive internal states of the computation. Under certain attack models, the number of side-channel traces that need to be collected is exponential in the number of shares. It is pertinent to consider how many traces (e.g., up to several million) a feasible adversary is expected to be able to collect, in order to successfully perform, for example, a partial-key recovery. E31: C4.7 E32: C4.9

Distributed random number generation.

4.2.1.2 Distributed random number generation. Randomness is fundamental for masking techniques. If only one randomness source is available, then that becomes an attackable single-point of failure. Therefore, there is interest in exploring circuit implementations that are able to leverage multiple on-chip sources of randomness and combine them in a threshold manner.

Others. It is

This publication is available free of charge from: https://doi.org/10.6028/NIST.IR.8214A

4.2.1.3 Others. As a use-case for threshold circuit design, the initial phase of this project is comparatively more focused on AES. Nonetheless, it is foreseeable that the insights gained in developing guidelines for the implementation and validation of threshold circuit designs for AES may also be applicable to other symmetric-key cryptographic algorithms, e.g., a hash-based message authentication code (HMAC). Public-key cryptography is also implemented in single devices, but as a use-case for threshold circuit design we are comparatively more focused on AESand for some particular schemes, it is possible to enhance side-channel resistance by using masking-type techniques. For example, an approach may be to ensure that some repeated operation (e.g., an exponentiation) sensitive to side-channel attacks (e.g., timing attacks) does not operate on the actual secret as input. E33: C7.24

4.2.2 More complex cases

Actively secure AES enciphering.

4.2.2.1 Actively secure AES enciphering. Beyond passive security, it is desirable to develop resistance against combined attacks (side-channel and injected faults). This may An active adversary may, for example, be able to inject a certain number of faults (e.g., controlled or random value in a certain number of controlled locations) in some data unit (e.g., bit or byte), and may be able to collect numerous execution traces (e.g., up to several million). Thwarting these attacks may involve more sophisticated techniques , (e.g., producing and distributing correlated randomness, and verifying it), and istherefore considered as , therefore, considered more complex. Ways of achieving this include cryptographic checksums (such as message authentication codes), whose result cannot be predicted by an adversary with only a partial view of the internal state. To be pertinent these schemes should be demonstrably better than a simple redundant execution of the circuit computation. E34: C4.8

This publication is available free of charge from: https://doi.org/10.6028/NIST.IR.8214A

5 Features of standardization items

The previous section enumerated several examples of possible standardization items at a high-level (domain–primitive–mode). However, an actual process of standardization will require taking into consideration factors such as validation suitability (~~§5.1~~Section 5.1), configurability and security features (~~§5.2~~Section 5.2), and modularity (~~§5.3~~Section 5.3).

5.1 Validation suitability

The process of standardizing new threshold schemes entails devising corresponding testing and validation requirements, which may differ from those for conventional implementations. This applies both to validation of modules and validation of the algorithms therein. Therefore, the E35: C2.2
validation framework should be looked at to consider which/whether extensions may be useful to accommodate a feasible validation of implementations of threshold schemes. Ideally, the proposed test E36: C9.4
and certification procedures should integrate with hardware and software development processes, clarifying which security levels they achieve.

Validation of modules. FIPS 140-2 and FIPS 140-3 (~~a.k.a.~~ similarly, ISO/IEC 19790:2012(E)) are security standards for cryptographic modules. They mandate the use of NIST-approved cryptographic primitives, referenced in Annexes to these standards in the cryptographic modules validated under them. The testing of the algorithm primitives is delegated to the Cryptographic Algorithm Validation Program (CAVP) as a prerequisite for module validation. In addition, FIPS 140-3 introduces requirements for side-channel leakage testing in its Annex F. These requirements are particularly important for single-chip implementations of threshold-schemes for cryptographic primitives, especially for block ciphers ~~— see Section 4.2~~(Section 4.2).

Validation of algorithms. The CAVP is established by NIST to validate the algorithm primitives used in modules. The CAVP uses automated tests based on the known-answer testing methodology. These tests try to assess the correctness and robustness of the implementation with emphasis currently given to the former.

In a typical scenario, one of the two participating parties (the NIST validation server and the client with an algorithm implementation under ~~test~~testing) using the Automated Cryptographic Validation Protocol (ACVP) sends to the other the pre- and post-conditions for a specific test of an implementation of a cryptographic algorithm. The other party ~~then~~ performs the same test with the received pre-conditions on an independently developed implementation of the same algorithm and verifies that the post-conditions are the same. Going forward, the CAVP is working on enhancing the depth and coverage of algorithm tests to cover a bigger portion of the security assertions contained in any of the cryptographic primitive standards ~~,~~ (e.g., digital signatures (FIPS 186) ~~,~~ and AES (FIPS 197)~~, etc~~).

5.2 Configurability and security features

Some ~~detailed configuration~~ important configuration details and security features need to be considered in the phases of defining criteria for calls for contribution, and their evaluation/comparison. ~~Some~~

This publication is available free of charge from: https://doi.org/10.6028/NIST.IR.8214A

of them These details and features may also depend on more detailed application scenariosto choose as motivation. We describe some important aspects here. the considered application scenarios.

5.2.1 Threshold numbers

We typically consider thresholds based on k-out-of-n Shamir secret sharing, possibly with variable k and n across the lifetime of the scheme. The n-out-of-n case with static n may also be relevant, when significantly more efficient. It is important to identify the proportion of dishonest parties (e.g., dishonest minority, all-but-one dishonest) that is allowed for each security property of interest, and whether threshold values are static or dynamic. For example, threshold schemes are typically based on some kind of k-out-of-n secret sharing, possibly with variable k and n across the lifetime of the scheme. The parameter k may imply different thresholds for different properties. Particular cases may also be relevant, such as the special case of n-out-of-n case with static n, especially when significantly more efficient.

5.2.2 Rejuvenation of components

In several application settings of threshold schemes, the ability to support the rejuvenation of components is essential. Rejuvenations can be proactive or reactive, and parallel or sequential. In the For reactive rejuvenations the system needs to be capable of some kind of intrusion detection. In particular, the recovery may be more efficient if the detection is accompanied by the ability to identify the misbehaving components. In the multi-party domain, a rejuvenation may include an actual replacement of a physical machine, or the rebooting of a virtual machine , and may include onboarding the state of the new componentwith corresponding onboarding of its state. In the single-device setting this may involve redoing a secret sharing of an encryption key. Forward secrecy is one property of interest that can be related to rejuvenations. In some settings, past actions may remain secured (e.g., with respect to confidentiality) even if the threshold assumption is broken at a point in time. E37: C3.5 E38: C5.2

5.2.3 Advanced security properties

A meaningful assertion of security for a threshold scheme depends greatly on the applicability of the underlying model, on the environmental conditions in which a scheme is implemented, and on what happens when assumptions are violated. Therefore, when devising, evaluating, and comparing possible threshold schemes for standardization, it is important to consider to what extent the schemes need to satisfy certain properties , such as:and/or have new requirements.

- (**Composability**). in which In what way does security remain when the scheme is composed with other protocols, including in concurrent executions, possibly depending on the actual instantiation of a required trusted setup?
- **(Adaptive security)**is Adaptive security. Is the adversary allowed to observe the protocol execution before deciding which components to corrupt?
- **(Graceful degradation)**is Graceful degradation. Is there a controlled vs. uncontrolled breakdown as soon as the threshold number of corruptions is surpassed?

This publication is available free of charge from: https://doi.org/10.6028/NIST.IR.8214A

- ~~**(New properties)**~~Termination options. How is the scheme characterized in terms of termination, e.g., with respect to fairness, guaranteed output delivery and identifiable abort? E39: C2.3
- Cryptographic assumptions. Of the cryptographic and/or setup assumptions required by the threshold scheme, which ones (if any) differ (e.g., stronger and/or incomparable) from those of the original scheme? It is important to assess how the potential break of assumptions may jeopardize security in comparison with the original (1-out-of-1) scheme. E40: C1.2, C3.6, C5.3
- New properties. The set of security properties to be required from threshold schemes can be more complex than with the corresponding conventional schemes, and may require some redefinition. For example, in an indistinguishability game for decryption, one may have to count adversarial queries made by isolated components, even if ~~such component is~~ those are then not part of an actual decryption. As another example, in a multi-signature scheme one needs to require the infeasibility of finding two distinct sets of secret keys that yield the same combined public verification key.

Other aspects. All aspects of implementation and application should be considered carefully when proposing the use of a threshold scheme. For example, a relevant concern in the execution of cryptographic primitives is determining the allowed scope of use of a secret key, such as whether it may be limited to signing only vs. decryption only. E41: C11.14

5.3 Modularity

The ~~process of~~ complex process of developing standards for threshold schemes can benefit from a modular approach at diverse levels. Upcoming standards should have the ability to share commonalities, and be flexible to enable appropriate solutions in a context of continuous innovation. Optimally, their building blocks can also be a useful basis for subsequent developments, without detriment to the effectiveness and credibility of each standard during its validity period. The process of standardizing multiple threshold schemes should consider appropriate tradeoffs of construction complexity (from building blocks to complex compositions) and specification detail (from security definitions to concrete instantiations). Figure 3 represents the abstract states and alternative paths of the evolution process, ~~towards~~ toward obtaining standardized threshold schemes that are concrete and provably secure instantiations of compositions of ~~well understood~~ well-understood building blocks. The figure shows four symbolic quadrants, explained ~~ahead~~below. E42: C1.5, C8.4

5.3.1 Security definitions of building blocks (Q_1)

Reference definitions of abstract gadgets (e.g., ~~such as~~ secret sharing and commitment schemes) can be reused across various threshold schemes, promoting interoperability and alleviating redundant redefinitions. This allows a more modular/compositional description of complex protocols. When incorporating ~~for the first time~~ a gadget into a standard for the first time, the gadget should have a ~~well defined~~ well-defined interface specified in that standard. This makes it possible that future standards refer to such descriptions based only on the corresponding interface and security properties. Some other examples of gadgets may include *consensus*, *generation of correlated randomness*, *reliable broadcast*, *oblivious transfer*, and *garbled circuits*. Their treatment as modules alleviates the burden

This publication is available free of charge from: https://doi.org/10.6028/NIST.IR.8214A

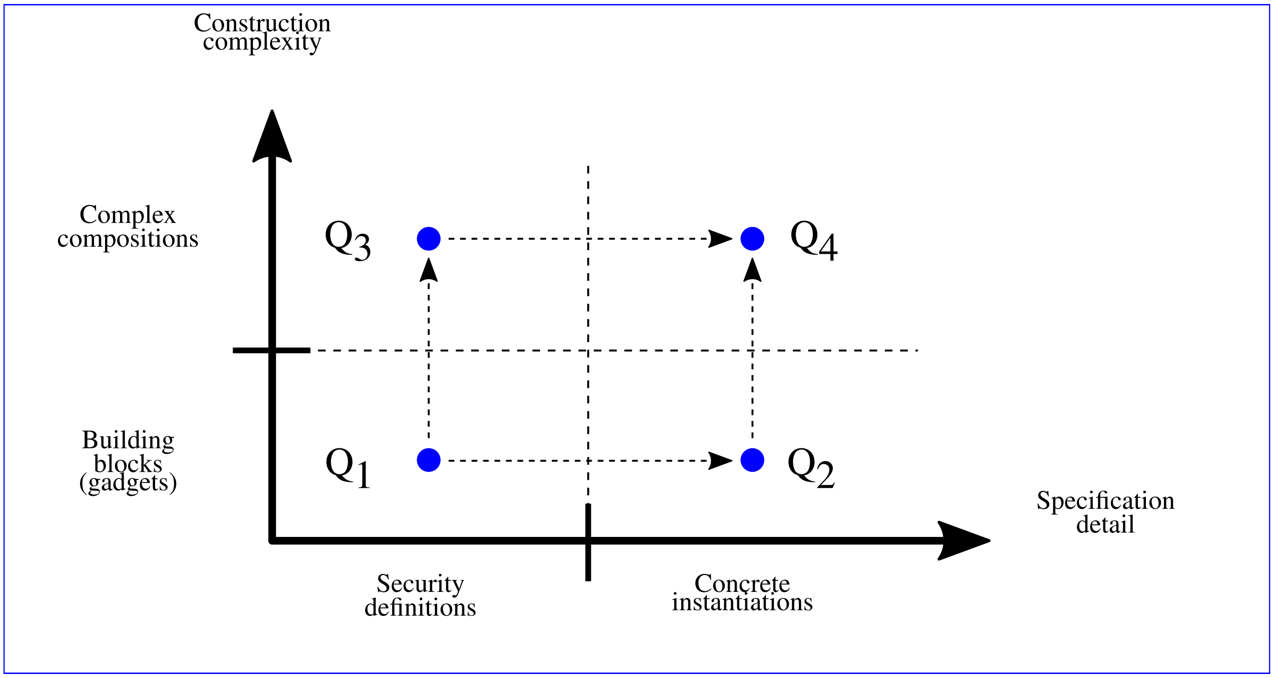


Figure 3. Modularity tradeoffs

of compiling from scratch arguments about the security of a more complex concrete protocol based on them, provided that composability properties are taken ~~in~~ into consideration.

In a similar vein, it can be useful to identify the basic arithmetic operations (e.g., modular exponentiation, ECC point multiplication and generation of random numbers) that may be common across multiple threshold schemes (e.g., within DLC and IFC), as well as the techniques to implement them in a way that offers some resistance to side-channel attacks. E43: C7.25, C7.27

Secret sharing is a particular case of a gadget applicable across all primitives. ~~Assuming~~ On its own it can also be useful to facilitate policies regarding separation of duties. Additionally, assuming that a key has been secret shared, some simple threshold schemes follow in a straightforward manner, using techniques very similar to the original algorithm. ~~Conversely,~~ Standards for more complex threshold schemes ~~are likely to benefit from reference definitions of other gadgets~~may also rely on other gadgets, and thus benefit from corresponding reference definitions, since they may be substantially different from the baseline cryptographic primitive being thresholdized. E44: C7.13 C7.19

5.3.2 Concrete instantiations of building blocks (Q_2)

The optimized low-level specification of a gadget, such as a commitment scheme, can vary across concrete protocols. Useful guidance may ~~thus consider comparing~~ compare concrete constructions of gadgets applicable across various threshold schemes. For example, for commitment schemes one ~~can~~ could devise guidance on how to implement hash-based commitments and Pedersen commitments, and in which cases each may be preferable, based on comparative advantages. In the single-device setting, these gadgets may correspond to components useful in a circuit, such as an AES S-box and E45: C4.6

This publication is available free of charge from: https://doi.org/10.6028/NIST.IR.8214A

finite-field multipliers.

Certain building blocks are in a peculiar position of having very well understood properties, and being widely useful, but not having standardized a version suitable to certain threshold schemes. For example, the current set of NIST standardized hash functions does not include those hash functions referred to as friendly for multi-party computation (MPC-friendly), which — compared with SHA-2 and SHA-3 — would significantly improve by orders of magnitude the efficiency of distributed computation of a hash. This begs the question of whether some non-standardized versions of building blocks, if determined as the best for use in a future standard of threshold scheme, should first be developed as an independent standard, or be (initially) specified as a module inside of a broader threshold-scheme standard that uses said building block. E46: C2.5, C6.2

5.3.3 Security definitions for complex compositions (Q_3)

We want to take advantage of the clarity provided by There are significant advantages of clarity offered by the specification of ideal functionalities, or by a defined interface and comprehensive set of security properties. These can be used for defining the threshold modes being sought , and the properties that the corresponding protocols need to satisfy. However, they are They are, however, not the final goal in terms of standardization, but only a logical abstraction on the way.

5.3.4 Concrete instantiations of complex compositions (Q_4)

For each threshold functionality (Q_3) identified as being of interest for standardization, we want to eventually specify a concrete threshold scheme (Q_4) should eventually be specified. This should be describable as a composition of building blocks (Q_1) that are, as much as possible (without compromising security and efficiency), interoperable across different threshold schemes, even under different instantiations (Q_2).

This publication is available free of charge from: https://doi.org/10.6028/NISTIR.8214A

6 Development phases

This section discusses the possible development phases ~~towards~~ toward standardization, putting special emphasis ~~of~~ on the types of calls for contributions that they may entail. ~~We seek~~ The goal is to have a transparent and open process, involving the community of stakeholders [NISTIR 7977].

~~We define four generic *phases* towards~~ The following discussion is in context with the structure given in the preceding part of this document, including the organization of the high-dimensional space of potential threshold schemes for standardization (Section 2), a consideration of motivating applications (3), the initial high-level identification of possible standardization items (Section 4), and various important features (Section 5).

The upcoming process for new standards of threshold schemes is envisioned in four phases:

1. ~~**Roadmap**~~Criteria**.** ~~**Develop a preliminary roadmap (including discussion of this document).**~~ Develop differentiated criteria suitable for various foreseen standardization items. E47: C11.15
2. **Calls.** ~~Devise~~ Perform calls for contributions, ~~with timelines and criteria for evaluation of input~~based on criteria and timelines.
3. **Evaluation.** Obtain and evaluate ~~contributions provided upon a call~~input obtained in the context of calls for contributions.
4. **Publish.** Write and publish new standards and guidelines

~~After settling on the preliminary roadmap, the~~ While most of the preparatory phase has been common to the two tracks, and has addresses the primitives generically, the definition of criteria and the subsequent phases should be tailored independently for each track, and possibly per identified standardization item, ~~with separate~~ possibly with distinct timelines. For some items, some phases may have several rounds, ~~e.g.,~~ such as possibly alternating several calls for contributions (phase 2) and ~~evaluation of~~ corresponding evaluations (phase 3)~~contributions~~.

Each phase is composed of three sub-phases (possibly with several internal rounds):

a. ~~produce~~ Produce draft documentation and call for feedback~~;~~.

b. ~~evaluate~~ Evaluate and integrate external feedback~~;~~.

c. ~~publish~~ Publish documentation.

6.1 Phase 1 — Develop ~~a preliminary roadmap~~criteria

~~The main goal of the initial phase (and of this document) is to provide a structured approach (Sections 2 and 3) for tackling the high-dimensional space of potential threshold schemes for standardization. This allows an initial identification of possible standardization items (Section 4), at a high level, with some discussion on several paths to follow concurrently. The roadmap also identifies important features (Section 5) to be considered down the line, to be further specified in subsequent phases.~~

6.2 ~~Phase 2 — Develop criteria~~

E48: C11.15

This publication is available free of charge from: https://doi.org/10.6028/NISTIR.8214A

The NISTIR 8214 has already enumerated several representative questions to consider when reflecting ~~about criteria. To recall, here aresome to consider~~on criteria. Some of the relevant aspects are:

1. ~~definition~~ Definition of system model and threat model~~;~~.
2. ~~description~~ Description of characterizing features~~;~~.
3. ~~analysis~~ Analysis of efficiency and practical feasibility~~;~~.
4. ~~existence~~ Existence of open-source reference implementations~~;~~.
5. ~~concrete~~ Concrete benchmarking (threshold vs. conventional; different platforms)~~;~~.
6. ~~detailed~~ Detailed description of operations~~;~~.
7. ~~example application scenarios;~~ Example application scenarios.
8. ~~security~~ Security analysis (see also Section 5.2)~~;~~.
9. ~~automated~~ Automated testing and validation of implementations (see also Section 5.1)~~;~~.
10. ~~disclosure~~ Disclosure and licensing of intellectual property.

The above items are important factors to take ~~in consideration ,~~into consideration but are not themselves a specification of criteria. ~~In fact,~~ It is important to obtain further feedback about these items, and several of them ~~should remain as useful topics of future discussion, besides being recalled here for the purpose of soliciting feedback about them.~~ represent useful topics for future discussions.

Several of these items also encompass various sub-factors. For example, with respect to efficiency and practical feasibility, there are numerous metrics to benchmark, depending on the setting. In a single-device setting, it can be relevant to consider circuit area, number of clock cycles, frequency, and number of required random or pseudorandom bits, among other possibilities. Also from an adversarial point of view, it can be relevant to assess what limitations exist with respect to the rate of possible collection of traces, namely compared with feasible rates of re-keying. In a multi-party setting, one should consider the overall communication between parties, the round complexity, and the computational resources per party. The use of a reference platform(s) may also be beneficial when comparing various techniques.
E49: C3.4
E50: C4.3
E51: C9.3
E52: C3.4
E53: C4.5

The goal of phase ~~2~~ 1 is to issue criteria ~~,~~that are refined per standardization *item*. However, such criteria will only emerge after consideration of feedback from stakeholders, and may happen with different timelines for different items. Furthermore, certain aspects have a life span that goes beyond the initial (future) issuance of criteria. This is~~for example~~, for example, the case of performing benchmarks, collecting reference implementations developed by the community, and developing testing and validation procedures. The development of these continues after the selection of concrete threshold schemes in subsequent phases.

Section 7 adds more notes about expected feedback useful for a reflection on criteria.

6.2 Phase ~~3~~ 2 — ~~Collect and evaluate~~ Issue calls for contributions

E54: C11.15

The word "contributions" has a broad meaning. The type of expected contributions can significantly vary with the technical difficulties associated with the intended standardization item. Based on this, ~~we envision~~ different initial types of calls ~~(here described at~~ are envisioned (and described here at a high level):

This publication is available free of charge from: https://doi.org/10.6028/NIST.IR.8214A

1. Simpler cases: proposals for new standards or guidelines;.
2. More complex cases: preliminary exploration and reference descriptions/implementations.
3. Out of scope of standardization: new research contributions.

For some simple items, as well as for simple gadgets (e.g., secret sharing), a contribution call may simply ask for complementary feedback on a base scheme proposal by NIST. Some simple items may nonetheless also involve an actual call for proposals of threshold schemes. We do not envision these cases These cases are not being envisioned as *competitions*, as it is more likely that different proposals share common features and we may want it may be desirable to adapt features for some final protocols.

The technically more more technically challenging items may require complex choices about their internal gadgets and their composition. The process must enable an adequate evaluation and selection across a wide span of possible protocols for the same intended functionality. In this case, a multi-stage contribution process is appropriate, starting with a request for information and progressing to concrete protocol proposals over time.

We are also interested There is also interest in research results about useful threshold schemes that are out of scope for this standardization effort. For the multi-party setting, this includes schemes for post-quantum public-key encryption (i.e., their decryption and key-generation algorithms) and signatures. For the single-device setting, this may conceivably include schemes for threshold enciphering, authenticated encryption with associated data (AEAD), and/or hashing related to lightweight cryptographic schemes being currently evaluated. However, this interest does not imply a direct interest for corresponding new standards.

6.3 Phase 3 — Evaluation of contributions

The process must enable an adequate evaluation and selection across a wide span of possible protocols for the same intended functionality. In this case, a multi-stage contribution process may be appropriate, starting with a request for information and progressing to concrete protocol proposals over time. It is important that the process itself has a pace that enables a proper review by the public, including the participation of stakeholders (in particular cryptography experts) to scrutinize the presented proposals. E55: C11.15 E56: C1.3

We will try It is a priority of this project to engage with the research community in some appropriate manner structured manners (e.g., dedicated workshops), to keep informed about the state-of-the-art in the corresponding fields, and to converge to solutions whose soundness is widely accepted.

6.4 Phase 4 — Publish new standards

The process of developing and adopting new standards will take into consideration the possible options and corresponding security evaluations. This includes soliciting public contributions corresponding public feedback from external stakeholders.

In some cases, a simple addendum to an existing standard may be sufficient to define the new mode or modes of threshold operation. For example, for some threshold circuit designs, the standardization of the technique may correspond to defining guidelines with implementation requirements

This publication is available free of charge from: https://doi.org/10.6028/NIST.IR.8214A

to achieve certification at some security level. For other items, the standardization may result into a new standalone standard.

Ideally, the upcoming standards will be clear and instructive, and they will serve as an aid for developing secure system designs. In any case, the goal of achieving upcoming standards of threshold schemes is that they are useful on arrival (rather than obsolete), and enhance the security E57: C7.20
of the implementation of the corresponding cryptographic primitives, namely with respect to a wide range of side-channel and fault-injection attacks. E58: C9.2

This publication is available free of charge from: https://doi.org/10.6028/NIST.IR.8214A

7 Collaboration with stakeholders

As an immediate ~~followup to this roadmap, we want to solicit~~ follow-up to this document, it is necessary to gather specific feedback on the criteria for subsequent calls for contributions. To this effect, it is important to obtain feedback from stakeholders about the security definitions and interfaces (and/or ideal functionalities) (see Q_3) upon which protocols/techniques should be evaluated.

~~We value~~ NIST values the expert technical feedback from stakeholders and ~~will incorporate it in our~~ that feedback will be incorporated it in the standardization process. Along the way, future NIST Threshold Cryptography Workshops (NTCW) may constitute an essential way to obtain interactive public feedback. This can be a place to discuss evaluations about contributions made thus far within the standardization process, while covering a variety of approaches across the different domains, and considering distinguished features of interest across various items. Overall, the standardization E59: C1.4
process itself needs to lead the upcoming standards to be of high quality.

~~Section ??~~ Section 6.1 has already mentioned important elements ~~for which we expect useful feedback as collaboration~~of desired feedback from stakeholders. The following subsections enumerate a few further important aspects, as ~~we move towards~~ the process progresses toward issuing criteria for new threshold schemes in each domain.

7.1 Multi-party setting

~~We are interested~~ There is interest in the development of multi-party threshold schemes that improve key-confidentiality, ~~and~~ as well as operational integrity and availability for the implementation of cryptographic primitives~~of interest~~. It is relevant to:

1. Enumerate useful threshold ~~-~~modes of operation.
2. For each intended mode, define the intended ideal functionality (and identify corresponding possible trusted setups) and/or game-based security definitions.
3. Identify main security properties to be derived from ideal functionalities when their trusted setups are bootstrapped in concrete settings and with concrete techniques.
4. Enumerate the gadgets whose reference definition is useful (as well as definitions already present in other standards).

7.2 Single-device setting

~~We are interested~~ There is interest in the development of threshold circuit designs that improve resistance against side-channel attacks and/or fault attacks in the single-device domain. It is relevant to:

1. Enumerate and define the desirable properties (e.g., uniformity ~~,~~and non-completeness~~, ...)~~) that are possible to achieve in threshold circuit designs.
2. Identify useful construction paradigms for threshold circuit design and ~~identify~~ the gadgets that are useful to implement them.
3. Indicate the models/conditions under which the threshold schemes may enable a higher resistance to side-channel and/or fault attacks ~~,~~(e.g., quantifying the increase in the number of traces required for a successful differential power analysis attack).

This publication is available free of charge from: https://doi.org/10.6028/NIST.IR.8214A

4. Indicate possible parameters (e.g., masking order ~~,~~ and number of shares) for realistic implementations of threshold circuit designs.

This publication is available free of charge from: https://doi.org/10.6028/NIST.IR.8214A

References

[ACVP] National Institute of Standards and Technology (2019). *Automated Cryptographic Validation Protocol*. GitHub:usnistgov/ACVP, https://github.com/usnistgov/ACVP.

[CAVP] National Institute of Standards and Technology. *Cryptographic Algorithm Validation Program*. https://csrc.nist.gov/projects/cryptographic-algorithm-validation-program. Accessed 2019.

[FIPS 140-2] National Institute of Standards and Technology (2001). *Security Requirements for Cryptographic Modules.* (U.S. Department of Commerce, Washington, D.C.) Federal Information Processing Standards Publication (FIPS PUBS) 140-2. Change Notice 2 — December 03, 2002. DOI: 10.6028/NIST.FIPS.140-2.

[FIPS 140-3] National Institute of Standards and Technology (2019). *Security Requirements for Cryptographic Modules.* (U.S. Department of Commerce, Washington, D.C.) Federal Information Processing Standards Publication (FIPS PUBS) 140-3. DOI: 10.6028/NIST.FIPS.140-3.

[FIPS 186-5] National Institute of Standards and Technology (2019). *Digital Signature Standard (DSS).* (U.S. Department of Commerce, Washington, D.C.) Draft Federal Information Processing Standards Publication (FIPS PUBS) 186-5. DOI: 10.6028/NIST.FIPS.186-5-Draft.

[FIPS 197] National Institute of Standards and Technology (2001). *Advanced Encryption Standard (AES).* (U.S. Department of Commerce, Washington, D.C.) Federal Information Processing Standards Publication (FIPS PUBS) 197. DOI: 10.6028/NIST.FIPS.197.

[NISTIR 7977] Cryptographic Technology Group (2016). *NIST Cryptographic Standards and Guidelines Development Process*. (U.S. Department of Commerce, Washington, D.C.) National Institute of Standards and Technology Internal Report (NISTIR) 7977. DOI: 10.6028/NIST.IR.7977.

[NISTIR 8214] Luís T. A. N. Brandão, Nicky Mouha, and Apostol (2019) Vassilev. *Threshold Schemes for Cryptographic Primitives: Challenges and Opportunities in Standardization and Validation of Threshold Cryptography*. (U.S. Department of Commerce, Washington, D.C.) National Institute of Standards and Technology Internal Report (NISTIR) 8214. DOI: 10.6028/NIST.IR.8214.

[NTCW 2019] National Institute of Standards and Technology (2019). *NIST Threshold Cryptography Workshop*. https://csrc.nist.gov/Events/2019/NTCW19. Gaithersburg, March 11–12.

[Proj-LWC] National Institute of Standards and Technology (2020). *Lightweight Cryptography Project*. https://csrc.nist.gov/projects/Lightweight-Cryptography.

This publication is available free of charge from: https://doi.org/10.6028/NIST.IR.8214A

[Proj-PEC] National Institute of Standards and Technology (2020). *Privacy-Enhancing Cryptography Project*. https://csrc.nist.gov/projects/Privacy-Enhancing-Cryptography.

[Proj-PQC] National Institute of Standards and Technology (2020). *Post-quantum Cryptography Project*. https://csrc.nist.gov/projects/post-quantum-cryptography.

[Proj-TC] National Institute of Standards and Technology (2020). *Threshold Cryptography Project*. https://csrc.nist.gov/projects/Threshold-Cryptography.

[SP 800-56A] Elaine Barker, Lily Chen, Allen Roginsky, Apostol Vassilev, and Richard Davis (2018). *Recommendation for Pair-Wise Key-Establishment Schemes Using Discrete Logarithm Cryptography.* (National Institute of Standards and Technology, Gaithersburg, MD) NIST Special Publication (SP) 800-56A Rev. 3. DOI: 10.6028/NIST.SP.800-56Ar3.

[SP 800-56B] Elaine Barker, Lily Chen, Allen Roginsky, Apostol Vassilev, Richard Davis, and Scott Simon (2019). *Recommendation for Pair-Wise Key-Establishment Using Integer Factorization Cryptography.* (National Institute of Standards and Technology, Gaithersburg, MD) NIST Special Publication (SP) 800-56B Rev. 2. DOI: 10.6028/NIST.SP.800-56Br2.

[SP 800-90A] Elaine Barker and John Kelsey (2015). *Recommendation for Random Number Generation Using Deterministic Random Bit Generators*. (National Institute of Standards and Technology, Gaithersburg, MD) NIST Special Publication (SP) 800-90A, Rev. 1. DOI: 10.6028/NIST.SP.800-90Ar1.

[SP 800-186] Lily Chen, Dustin Moody, Andrew Regenscheid, and Karen Randall (2019). *Recommendations for Discrete Logarithm-Based Cryptography: Elliptic Curve Domain Parameters*. (National Institute of Standards and Technology, Gaithersburg, MD) Draft NIST Special Publication (SP) 800-186. DOI: 10.6028/NIST.SP.800-186-draft.

This publication is available free of charge from: https://doi.org/10.6028/NISTIR.8214A

A ~~Application use cases~~

Appendix A — Application use cases

~~In this section we describe~~ This section describes at a high level several conceivable applications that take advantage of threshold schemes for cryptographic primitives. This is intended ~~as~~ to be an aid to identify, motivate and select concrete items of interest for standardization.

A.1 Single-device encryption resistant to side-channel attacks

The hardware implementation of cryptographic algorithms has gained a significant and growing stake in the industry. Large amounts of sensitive data are now processed in hardware, which creates the need for faster implementations. Most semiconductor manufacturers have incorporated dedicated hardware accelerators for cryptography that perform orders of magnitude faster than software implementations. ~~Even though asymmetric algorithms~~Algorithms of asymmetric cryptography, such as RSA and ~~even~~ ECC digital signatures, can be implemented by a hardware accelerator, ~~in order~~ as a way to reduce the processing time of private key operations. However, these algorithms are sometimes not suitable for severely constrained devices in the Internet of Things (IoT), due to the significant resources required, which results in low performance on such platforms. ~~As a result~~Indeed, many IoT devices have only hardware engines for symmetric cryptography primitives, such as AES.

At the same time, conventional hardware implementations of cryptographic algorithms have created significant problems in terms of side-channel leakage. Traditional techniques for leakage mitigation are costly and ad hoc~~. Such~~ , and such implementations are also susceptible to fault attacks. ~~In this context we ask~~Thus, a question arises: *what type of algorithm is the most widely used in hardware and stands to gain the most from a standard mechanism for mitigating leakage and/or fault attacks, if threshold schemes for it are developed and standardized?*

Symmetric-key cryptographic algorithms, such as block ciphers and message authentication codes, tend to be difficult to protect~~,~~. Furthermore, the leakage pattern of hardware implementations ~~of~~ is vastly different from what emanates from software implementations. Glitches and other physical effects result in stronger leakage for hardware implementations of symmetric cryptographic algorithms (compared to software ones). Based on this, for the single-device track ~~we propose to focus~~ there is value in focusing on hardware implementations of block-cipher algorithms (AES strongly preferred) and ~~develop~~ in developing standards for threshold schemes to mitigate the risks of side-channel leakage and/or fault attacks.

A.2 Protection of secrets at rest

Most cryptographic applications involve a secret, which if revealed to an adversary results in a security failure. For example: a secret key corresponding to a public certificate can decrypt encrypted messages whose content was intended only for the key owner; a secret key from a crypto-currency can be used to spend the original funds of the owner; the secret signing key of a certificate authority (CA) can sign certificates as the CA. The key ~~also needs to~~ must also be available to the legitimate user — losing the key may imply losing a digital identity, in the case of a signing key, or losing

This publication is available free of charge from: https://doi.org/10.6028/NIST.IR.8214A

access to funds, in the case of a crypto-currency private key.

In any of the ~~above~~ mentioned cases, the storage of the secret key in one place represents a *single point of failure* for confidentiality, integrity, or availability. This can be mitigated by using secret sharing to distribute, across multiple parties, the trust in the storage of secrets. ~~Example use-cases: a CA where the signing key is~~ In one use-case, a CA may have its signing key secret-shared among several employees, such that no single employee alone has access to the key~~;~~. In another use-case, a "social backup" system for crypto-currency wallets ~~, whereby the user distributes~~ may allow the user to distribute shares of the key to several friends, such that if the ~~user's~~ device of user device is lost or breaks, the user can still recover the key from the shares. Once a secret key is protected at rest using secret sharing, there are threshold schemes that enable avoiding reconstruction of the key even when the key needs to be used in some operation.

A.3 Confidential communication

For secure communications it is essential to ensure that secret messages are only decrypted by legitimate recipients. An attacker who steals Alice's secret decryption key can read messages intended for Alice. Threshold decryption can help protect confidentiality. It can~~for example~~, for example, be used across devices, analogously to multi-factor "authentication" for a single person, such that unauthorized parties (in this case hacked or stolen devices) cannot break the confidentiality of messages, without using multiple shares of the key. Similar considerations apply to ~~protection of authenticity of messages,~~ the protection of message authenticity, (i.e., preventing an attacker from masquerading as Alice to others, with respect to a secret signing key).

Using a threshold decryption (e.g., RSA) in a shared-O mode, the multiple parties compute separate shares of the decryption plaintext~~, and then~~. Then, a combiner (possibly the end recipient, i.e., the client) receives the shares and computes the plaintext from them. This mode of operation protects the secrecy of the (distributed) key (as a main feature) as well as the confidentiality of the decrypted message (as an added feature). In some settings this may provide a kind of accountability, since it requires the explicit participation of multiple parties, who can~~for example~~, for example, log their operations for future audits. Also, ~~in an enhanced~~ if the scheme is auditable ~~mode~~ then the recipient of the final decryption can verify which decryptor parties were involved.

A.4 Decentralization of trust for key generation and distribution

Key generation and distribution are essential phases of many cryptographic schemes and applications. For example, a key distribution center (KDC) can act as a trusted service that distributes symmetric secret keys to clients, to enable private communication within groups or to mediate access to other services. ~~A KDC thus~~ Thus, a KDC represents a single point of failure~~: if~~. If the KDC is offline, the clients cannot securely communicate ~~nor~~ or access needed services~~; if it~~. If the KDC is hacked, the attacker can learn the secret keys in use by clients, and can obtain tickets to access any services. The same considerations apply~~for example~~, for example, to an identity-based encryption scheme, where a trusted server holds the master key that is required to generate a new secret key for every new member (identity) in an organization. Yet another example is the use of a "dealer" as a trusted party generating a secret key (possibly with a complex structure, such as an RSA key), only to then

This publication is available free of charge from: https://doi.org/10.6028/NIST.IR.8214A

secret share it across multiple parties of a subsequent threshold signing of or decryption scheme.

To eliminate this single point of failure, a set of servers can jointly act as a KDC or dealer in a way that no individual server knows any of the secret keys, and so that services remain available as long as a certain threshold number of servers have not been hacked or taken offline. This threshold property can be based on distributed key generation and use of secret sharing, possibly with proactive and verifiable properties. The latter properties allow the servers to jointly refresh the secret shares (in order to recover from the potential compromise of some servers) and to ensure that their shares are consistent. The distribution of servers prevents any server from learning any master secret key, while the actual distribution of new keys may fit within a shared-O mode, so that no server learns any new secret key. For example, verifiable delay functions (which can be useful for various applications) can be based on an RSA public key whose corresponding secret key needs to be unknown to everyone. E60: C3.7

A.5 Accountability and prevention of ill-intentioned operations

Entrusting a single individual with the ability to decrypt or sign a message may invite foul play, if the result cannot be externally verified as correct or its computation does not require agreement between multiple parties.

For example, to authorize a large bank transfer, it can be useful to require agreement between several managers. A policy can state that transactions above a certain amount are only valid once after signed off by at least two out of three bank managers, to prevent the authorization of errant transfers intended by a single ill-intentioned manager. Certain threshold signature schemes (including multi-signatures) enable this in an a functionally interchangeable mode, such that the output is syntactically equivalent to an original signature— this . This property can be important for records where size matters (e.g., storage in a blockchain) and where the policy on the number of signers may be dynamic. If a single original signing key was secret-shared between the managers, then the bank can internally know that a large enough subset of managers got together, though possibly not knowing (from the signature itself) which ones. If a "multi-signature" scheme is used, then each manager can have its their own independent secret-public key pair, enabling an mode where it possible . This becomes auditable in the sense of allowing to check which managers participated, thereby facilitating accountability. The same consideration could be applied, for example, to an application use-case of notary services. E61: C7.18

Compared with a simple concatenation of signatures, certain concise threshold signatures (e.g., when the secret-key is secret-shared) may also be desirable as a feature of not exposing the identity of the signers and the corresponding organizational structure.

A.6 Distribution of trust across secure environments

Hardware security modules (HSMs) are often used to safeguard high-value secret keys. They perform cryptographic operations, such as signatures, only inside a hardened-security environment that attempts to prevent exfiltration of the keys. However, even HSMs are subject to new vulnerabilities and side-channel attacks that enable an insider attacker, with physical access to an HSM, to exfiltrate a signing key before the HSM is patched. To mitigate this attack, it is possible to use a diversity of HSMs as multiple parties in a threshold scheme.

This publication is available free of charge from: https://doi.org/10.6028/NIST.IR.8214A

For certain threshold schemes, such as for a threshold RSA signature, each HSM only has to perform an already supported cryptographic operation. Each HSM simply computes and outputs a regular RSA signature, using a signing key share, and then some external non-HSM device combines the output shares to obtain the final RSA signature. This application can be enabled by a dealer that, in an initial safe/protected phase, secret-shares the RSA key, and distributes one share to each HSM (across diverse locations). For more complex threshold schemes (including RSA key generation without a dealer), the threshold operations may require customized programing and interactions between parties. This can be achieved for example by diverse virtual machines running in various and diversified computers (e.g., with different operating systems and protected by different access control mechanisms).

A.7 Distributed password authentication

In a typical password-based authentication, a client sends its username and password to a server, via an encrypted channel, and then the server computes a salted hash of the password and checks the result against a verification table of hashes. This setting has several single-points of failure: (i) if the server fails, then the authentication service becomes unavailable; (ii) if the server's database is leaked by an intruder, then an attacker can use an offline "dictionary attack" to find which passwords in the dictionary match the database; also, and (iii) if the server is hacked with spyware, then the intruder may be able to read in real time the passwords sent by clients.

Without changing the underlying hash-based mechanism, the first two mentioned issues can be rectified by a simple threshold approach. Each salt in the verification table can be secret-shared across a set of n servers, such that any subset of f or fewer shares has no information about the not-in-use verification salts, and any subset of $f+1+a$ uncompromised servers (for some non-negative a) can reconstruct a verification (salted) hash when so requested. In this example, the enhanced confidentiality of the values stems from the threshold property of the threshold secret sharing, without using any encryption. The use of salts prevents the attacker from benefiting from pre-computations in the actual case where the verification table is leaked (if more than the threshold number of servers is compromised).

The online attack (issue iii above) can be addressed with extra steps, such asfor example: (i) the client sends the password in a shared-I mode ,(i.e., as separate secret shares to each server); (ii) then the servers, each also with a salt share, jointly compute the salted hash, but without even recombining the salt (efficiency-wise this may benefit from a hash function that is friendly with respect to distributed computations); and (iii) if the output matches the expected hash, then the user is authenticated. Thus, besides the secret-sharing of the input, the complexity of the operation lies only on the side of the servers.

The above description is meant for illustration purposes only. An actual consideration for a real authentication scheme with threshold properties would require a proper security analysis and would likely warrant further considerations. For example, other solutions exist to prevent the client from leaking any information about the password. Some of these solutions are implemented in practice in the space of password authenticated key exchange (PAKE), and their threshold variants could be performed using threshold versions of oblivious PRFspseudo-random functions. These can be

This publication is available free of charge from: https://doi.org/10.6028/NIST.IR.8214A

resilient against an active eavesdropper even if the client does not have an initial secure channel with the servers. However, some of these solutions go beyond the scope of the threshold modes currently defined in Section 2.3, since they require the client to actively participate in a secure computation, performing actions beyond secret sharingsecret-sharing.

Track changes

This page and the following are not part of the final NISTIR. This section explains changes between the draft NISTIR (dated 2019-05-03) and the final NISTIR. Each subsequent table contains a comment set, indexed as C*x* (where *x* is an integer). Each table has a header as follows:

#	Item id	**Comment set C*x***: *Name(s) of commentator(s)*	Notes and changes	Edit id

From left to right, the columns represent:

- **#**: A consecutive positive integer, used to count all described items of contribution
- **Item id:** An index of the comment item, subordinate to the table index. For example, C1.3 is the third item (row) of the comment set (table) C1.
- **Comment set C*x*:** The rows after the header contain the received public feedback comments. The header shows an identifier **C***x* (with integer *x*) of the set of received **c**omments, and the name(s) and affiliation of the person(s) who submitted the comments.
- **Notes and changes:** Notes replying to the comment; description of changes made to the document.
- **Edit id:** Index (or indices) of the **e**dits (E*y*, with integer *y*) made in the document. Across the document, changes will be marked on the right margin, with this index, so that the reader can hyperlink directly to the "notes and changes" description.

List of Comment Sets

Comment set C1 — Samuel Ranelluci (Unbound Tech, Israel) . . . 40
Comment set C2 — Jakob Pagter (Sepior) . . . 41
Comment set C3 — Tore Frederiksen (Alexandra Instituttet, Denmark) . . . 43
Comment set C4 — Svetla Nikova & Vincent Rijmen COSIC (KU Leuven, Belgium) . . . 46
Comment set C5 — Chelsea Komlo (University of Waterloo) . . . 48
Comment set C6 — Nigel Smart (COSIC – KU Leiven) . . . 49
Comment set C7 — Phillip Hallam-Baker (Venture Cryptography) . . . 50
Comment set C8 — Ventzi Nikov (NXP Semiconductors) . . . 61
Comment set C9 — Simon Hoerder & Elke De Mulder (Rambus Inc.) . . . 62
Comment set C10 — Ronald Tse, Wai Kit Wong, Daniel Wyatt, Nickolay Olshevsky, Jeffrey Lau (Ribose Inc.) . . . 64
Comment set C11 — Other editorial and content adjustments (from internal revision) . . . 71

#	Item id	**Comment set C1**: Samuel Ranelluci (Unbound Tech, Israel)	Notes and changes	Edit id
1	C1.1	**Introduction.** We would like to thank NIST for their effort towards standardizing threshold cryptography. Draft NISTIR 8214A is a well-written and thorough document and provides a good foundation towards this goal. I hope that the following considerations can be useful to NIST towards standardizing threshold cryptography.	– **NOTE:** Thank you for the encouragement and comments. – **CHANGED:** Added a note to the acknowledgments.	E8
2	C1.2	**Cryptographic Assumptions.** The draft does not explicitly mention cryptographic assumptions. When selecting which protocols to standardize, it is important to consider the underlying cryptographic assumptions needed to realize a given protocol. It may be the case that certain protocols are more efficient but rely on less-tested assumptions than other protocols that realize the same functionality. For example, the protocols described in [1, 2] implement threshold ECDSA but can only be instantiated from class groups of imaginary quadratic fields. When standardizing protocols, it is also important to state the cryptographic assumption on which this protocol is based. [1] Guilhem Castagnos, Dario Catalano, Fabien Laguillaumie, Federico Savasta, and Ida Tucker. Two-party ecdsa from hash proof systems and efficient instantiations. In Annual International Cryptology Conference, pages 191—221. Springer, 2019. [2] Guilhem Castagnos, Dario Catalano, Fabien Laguillaumie, Federico Savasta, and Ida Tucker. Bandwidth-efficient threshold ec-dsa. In IACR International Conference on Public-Key Cryptography, pages 266–296. Springer, 2020.	– **RELATED:** C2.3, C3.6, C5.1, C5.3, C7.10 – **NOTE:** It is essential to consider cryptographic assumptions when selecting threshold schemes for standardization. There is a preference for reliance on long-standing and well understood assumptions. It is important to compare the assumptions of the original scheme vs. those of the threshold scheme, and the possible implications of their differences. – **CHANGED:** Added a bullet item "cryptographic assumptions" in Section 5.2.3, to make it explicit.	E40
3	C1.3	**Protocol Maturity.** A protocol should not be standardized unless it meets an age requirement. If a protocol has a flaw, then by waiting a few years, there is a chance that the flaw will be discovered. In addition, it would reduce the chance that NIST will begin standardizing an algorithm that will likely be superseded by a better protocol. To demonstrate this point, I will focus on the task of distributed RSA key generation described in the draft. It is likely that the recent work on distributed RSA key generation [3] will be improved upon in the near future. Thus, we recommend that protocols should not be standardized until they have reached a certain age. [3] Tore Kasper Frederiksen, Yehuda Lindell, Valery Osheter, and Benny Pinkas. Fast distributed rsa key generation for semi-honest and malicious adversaries. In Annual International Cryptology Conference, pages 331–361. Springer, 2018.	– **NOTE:** The process of standardization itself needs to have a pace that enables a proper review by the public, including by cryptography experts. We are counting on the participation of stakeholders to scrutinize the proposals presented throughout the process and present public feedback. We do not suggest a concrete age requirement (number of years since proposal), but it is important that future proposals support themselves with proper evidence of credibility. As mentioned, sometimes a newer protocol is an improvement of an older one, which means older is not necessarily better. – **CHANGED:** In Section 6.3, added a sentence emphasizing the importance of public review.	E56

#	Item id	**Comment set C1**: Samuel Ranelluci (Unbound Tech, Israel)	Notes and changes	Edit id
4	C1.4	**Higher Threshold of Provable Security.** Much of the scientific literature on threshold cryptography was published in conferences. Unfortunately, conference papers are often reviewed in a limited period of time and the papers are often written under severe time and space constraints. Thus, to ensure that only secure protocols are standardized, it may be necessary to expand the description of the protocols and provide expanded and rigorous proofs for those protocols during standardization.	– **NOTE:** We agree that a paper publication in a conference is not sufficient guarantee of suitability for a standard. But what should be the "threshold"? As mentioned in Section 6.1, a "detailed description of operations" and a "security analysis" are important aspects to consider. The standardization process itself needs to be directed towards setting a high quality bar. In this process it will be useful to obtain contributions and specific feedback from stakeholder. – **CHANGED:** In Section 7, added one sentence reinforcing the intention of high quality standards.	E59
5	C1.5	**Issues with Standardization.** The purpose of standardization is to provide better security and provide certification that enables others to trust technology based on threshold implementations. However, standardization can also have downsides. The following story illustrates a problem that could occur. Imagine a cryptographer who wants to realize a given cryptographic task and discovers that the current fastest standardized algorithm is not fast enough. In that case, the cryptographer has no choice but to design a custom protocol for the given task. This leads to the question of how can the same cryptographer certify that his solution is secure. In the worst case, the cryptographer could not certify his solution as secure. In this case, the cryptographer could be forced to use an inferior solution that might be rejected by a user due to its inefficiency. In this case, standardization would have hindered innovation and prevented the deployment of threshold cryptography. We hope that standards are developed in a way to provides modularity and flexibility. Hopefully, we can avoid the pitfalls that may occur due to standardization.	– **RELATED:** C7.20, C8.4, C9.5 – **NOTE:** Continuous development plays an important role in Cryptography; standardization of cryptographic schemes also has an important role in diverse contexts. We certainly want eventual standards to be useful for already conceived applications (e.g., Appendix A describes a few imagined use-cases). Efficiency is one relevant consideration, among several dimensions. Ideally, standards would be useful even if/when they no longer are the most efficient one by a small margin; at the same time, we do not want standards that do not help resolve real-world problems. Enabling properties of modularity and flexibility can be useful for upcoming standards. Section 5.3 focuses on the aspect of modularity; the previous NISTIR (8214) also explicitly mentions flexibility (e.g., "An open question is deciding at what level each standard should be defined and which flexibility of parametrization they should allow"). – **CHANGED:** In Section 5.3, added text emphasizing the importance of modularity and flexibility in a context of continuous innovation.	E42

#	Item id	**Comment set C2**: Jakob Pagter (Sepior)	Notes and changes	Edit id
6	C2.1	On behalf of Sepior, Denmark, I would like to make the following comments for this document:	– **NOTE:** Thank you for your comments. – **CHANGED:** Added a note to the acknowledgments.	E8
7	C2.2	On behalf of Sepior, Denmark, I would like to make the following comments for this document: Regarding what to standardise we believe that it would be relevant to look at how to extend the FIPS 140 validation suite to encompass TC based solutions as well.	– **NOTE:** The development of new standards should take into account the validation context, namely the FIPS 140 framework, as expressed in Section 5.1. – **CHANGED:** Added a sentence emphasizing the need to look at the validation framework.	E35

#	Item id	**Comment set C2**: Jakob Pagter (Sepior)	Notes and changes	Edit id
8	C2.3	For that to happen it will necessary to standardise things like the security model (e.g. honest/dishonest majority, active/passive, fairness/abort/termination, pro-active security, ...).	– **RELATED:** C1.2, C3.6, C5.1, C5.3, C7.10, C8.2 C3.6, C5.1, C3.5 – **NOTE:** As acknowledged in Section 5.2, there are numerous security features to consider. Section 5.2.1 already considered aspects of honest/dishonest majority. Section 5.2.2 already considered aspects of pro-active / reactive rejuvenations. Section 5.3 also refers to the need to consider security definitions, such as those related to compositions. These aspects should be considered in the standardization process, even when they are not the final goal of a standard. Note that Section 1.1 already mentions the case of reference definitions, which may be useful while developing standards for threshold schemes. – **CHANGED:** In Section 5.2.3, added an item referring to termination options (e.g., fairness, g.o.d., abort)	E39
9	C2.4	Currently we do not believe it makes sense to standardise specific protocols (say for ECDSA) or building blocks/primitives (e.g. commitments or zk-proofs), as research is still very active in these areas.	– **NOTE:** As mentioned in Section 1.1, “attaining some standards will require first establishing a clear rationale to support concrete selections.” – **CHANGED:** In Section 1.1, added some text acknowledging the challenge of considering techniques that are in a context of active research.	E12
10	C2.5	Also, and of relevance in its own right, it would also be relevant to have TC friendly cryptographic primitives included in the NIST standards, e.g. hash function or modes of operation for AES which are efficient to evaluate using MPC.	– **RELATED:** C6.2 – **NOTE:** There are conceivable building blocks whose set of current standardized versions may not cover the optimal conceivable instantiations for certain purposes of distributed computation. We hope that along the process the community of stakeholder provides useful feedback about the building blocks that can facilitate various thresholds schemes for standardization. – **CHANGED:** Added text reflecting on how to possibly incorporate building blocks into larger standards.	E46

#	Item id	**Comment set C2**: Jakob Pagter (Sepior)	Notes and changes	Edit id
11	C2.6	**A separate – and more technical – comment:** The draft defines Not-shared-IO, Shared-I, Shared-O, and Shared-IO. There is however another 'mode' not mentioned. In shared-I/O/IO the client is involved in the specific TC protocol, which is tied to the secret sharing scheme employed by the protocol. But, there are solutions which ensure end-2-end privacy and integrity of the client input and output without the client have to know the specifics of the protocol/secret sharing employed. The client could, e.g., encrypt with a one-time pad which is removed by the MPC computation, and likewise the client might have a key for a one-time MAC so with a MAC valued added inside the MPC. In a sense this is somewhere in between non-shared and shared: the client cannot ignore that a TC protocol is used, but it does not need to be aware of the specific sharing schemes used. This gives a nice decoupling of the client application and the threshold protocol. An examples of such an approach is given by: https://eprint.iacr.org/2016/037	– **NOTE:** It seems to us that the mentioned approach (including the one referred to by citation) fits within the mode where the client needs to compute a secret-sharing (SS) of the input. The shared modes (e.g., shared-I and shared-IO cases) are not necessarily constrained to a simple Shamir SS scheme, and it is conceivable that they can involve integrity and confidentiality enhancements, as suggested. You mentioned "In shared-I/O/IO the client is involved in the specific TC protocol", followed by "But, there are solutions ... without the client have to know the specifics of the protocol/secret sharing employed." Note that the proposed shared-I/O modes do not imply that the client knows or is aware of any intricacy of the threshold scheme (e.g., some SMPC) implemented by the threshold components, apart from having a compatible interface, i.e.: • the MPC protocol needs to interpret the input shares received from the client in a shared-I or shared-IO mode; • and the client needs to know how to reconstruct the final output from the output shares, in a shared-O or shared-IO mode. In summary, compared with an interface of a non-threshold scheme, where a client simply handles a simple input (request of cryptographic operation) and output (receive result of cryptographic primitive), we conceive enhancements based on secret-sharing or reconstruction on the side of the client, as well as based on complementary communication protocols (e.g., use of TLS for communicating with any party). While the shared-I/O modes allow the client to produce input shares and/or reconstruct output shares for/from a SMPC between the components of the threshold entity, the document does describe in Section 2.3 a related limitation: the client should not be required to be an agent of a secure multiparty computation where it has to perform a sophisticated protocol with each component of the threshold entity. – **CHANGED:** In Section 2.3, edited some text to clarify the scope of what the shared-I/O modes cover and not cover.	E18

#	Item id	**Comment set C3**: Tore Frederiksen (Alexandra Instituttet, Denmark)	Notes and changes	Edit id
12	C3.1	I have read through the preliminary roadmap for threshold cryptography, NISTIR 8214A, with great pleasure. In particular I enjoyed your suggestion of abstraction of modes as explained in section 2.3. I think it is great that you are putting in so much effort towards standardizing advanced cryptographic primitives and I am sure this will help such finding their way into the "real world". In any case, I wanted to share some of my thoughts and comments about this roadmap. To be concise I have separated into a couple of different bullet points.	– **NOTE:** Thank you for the encouragement and comments. – **CHANGED:** Added a note to the acknowledgments.	E8
13	C3.2	Although auditability is an interesting and valuable feature, I believe that it is equally interesting, from a practical point of view, to have the cryptographic algorithms using public keys be agnostic to whether its corresponding private key(s) are used in a threshold setting. That is, I believe it is very valuable to allow full compatibility with already existing software solutions and NIST specifications, when it comes to those algorithms of threshold cryptographic schemes that do not need to be executed in a threshold manner. For example, if an RSA signature has been computed using a threshold scheme it should still be possible to verify this signature using any software that is able to verify a standard RSA signature without modification. This will fit very nicely with the goal already mentioned on line 245-250 and again on line 418-421 of thresholdization of NIST-approved primitives.	– **NOTE:** We have a generic goal of "interoperability" with existing standards. In particular, we have put emphasis in functional interchangeability, which seems to match the "compatibility" property you mentioned. Also, in this document we have considered auditability in the sense of enabling a client to prove to external parties that it obtained its output from a threshold execution. This can be compatible with the above mentioned compatibility. For example, this is the case of certain multi-signatures, where the signature can then be checked against a set of public-keys of the intervening components of the threshold scheme. This is also the case of a regular threshold scheme enhanced by metadata (e.g., a kind of certificate of thresholdization) such that the client can clearly identify the output of the original cryptographic operation and, if so needed, can take advantage of the additional certificate for auditability purposes. – **CHANGED:** The new Section 2.4 contains a significant text revision to explain better our intended concept of *interchangeability*. Now we explain functional equivalence as a special case of intercangeability. In Section 2.5, added text to clarify that the possibility of auditability is not intended in detriment of interoperability.	E23, E24

#	Item id	**Comment set C3**: Tore Frederiksen (Alexandra Instituttet, Denmark)	Notes and changes	Edit id
14	C3.3	Even though focus is on thresholdization already existing NIST primitives as mentioned on line 245-250 and again on line 418-421, I am wondering if it would be acceptable to simply use NIST primitives to make new threshold primitives. What I am concretely thinking about is the threshold setting of symmetric encryption, e.g. AES. As mentioned on line 627-632 it is hard to implement a threshold version of this without using something as complex as multi-party computation. However, in case we are not considering the single-device setting and depending on the mode of operation, there might be several ways of achieving something equivalent to threshold en/decryption by simply using secret sharing and local AES operations. Say for the setting where a user simply wants to use a set of servers for secure and fail-safe storage of sensitive data, it would, in the shared-IO model, be sufficient for it to simply secret-share its data to the servers. If the goal instead is oblivious encryption, simply using a threshold public key encryption scheme should achieve the same result in a more efficient way given how inefficient general, non-arithmetic MPC computations are. Thus more considerations of why achieving threshold versions of AES are desired and if those goals could be met by "easier" threshold solutions might be worthwhile.	– **RELATED:** C7.20 – **NOTE:** There are indeed conceivable applications/constructions where NIST-approved primitives can be used as a building block. Some stakeholders may use NIST-approved primitives in such way. Along with the example in the comment, an application use-case of storing confidential data can leverage the use of secret-sharing and local AES-based encryption. A relevant question is whether some of those constructions would benefit from standardization and whether that should fit or not in our current scope. In the current phase of the project we have a focused goal and motivation towards threshold schemes for NIST-standardized primitives, namely considering there are settings where the use of cryptography requires that it be based on NIST-standardized primitives, and correspondingly validated implementations. We want to enhance the standards with the possibility of threshold implementations of those primitives already in use. We think it is important to have this as the first focus. Having said this, we also think that some possibilities will become clearer as we move forward, for example, with using complementary building blocks (such as secret-sharing) in standards. As mentioned in Section 6.2, “We are also interested in research results about useful threshold schemes that are out of scope for this standardization effort.” – **CHANGED:** No change.	
15	C3.4	In relation to the questions raised about development criteria in Section 6.2, some things that might be of significant importance to consider during development is round complexity and interaction requirements between servers. Interaction is generally not needed for the threshold secret operation in RSA schemes, but is needed for threshold ECDSA schemes. When continuing to also consider threshold key generation in such a setting, round complexity can also become verify significant in relation to overall efficiency.	– **NOTE:** There are settings where round complexity is an important metric to assess efficiency and practical feasibility. – **CHANGED:** In Section 6.1, added text mentioning that several metrics should be considered, including “round complexity” as an example of concrete metric.	E49
16	C3.5	Also in relation to features and measures of security, it might be worth considering if there is a desire to identity malicious parties, or simply if the fact that a threshold operation fails is sufficient.	– **RELATED:** C2.3 – **NOTE:** Aspects of intrusion detection are important in the context of threshold schemes. In particular, the detection and identification of misbehaving parties is useful to enable selective reactive rejuvenation of the offending components. This may also be relevant for models of covert security, where the ability to get caught may to some extent act as a deterrent against malicious behavior. – **CHANGED:** In Section 5.2.2, we added a bullet point alluding to this distinguishing property.	E37

#	Item id	**Comment set C3**: Tore Frederiksen (Alexandra Instituttet, Denmark)	Notes and changes	Edit id
17	C3.6	Another aspect in relation to this, especially when threshold key generation is considered, is acceptable security assumptions on top of the assumptions implied by the underlying schemes. Furthermore, in relation to security definitions and ideal functionalities there can be quite a few pitfalls when considering threshold key generation. For example, for threshold RSA key generation it is almost always the case that keys end up being based on primes congruent to 3 mod 4 and that a few bits of these primes are leaked.	– **RELATED:** C1.2, C2.3, C5.1, C5.3, C7.4. – **NOTE:** In the mentioned example, where the resulting RSA key is induced to being a Blum integer, the key is consistent with the syntactical requirements of an RSA key but may have been subject to some limitations. It is important to understand how each threshold scheme affects (or not) the probability distribution of the output of the underlying primitive being computed, and assess whether the induced modifications are acceptable or can be mitigated. These aspects should be made very clear in any upcoming standards. – **CHANGED:** Mentioned as one example the case of RSA composites of primes equal to 3 mod 4. The revised version of the document includes a more clear distinction of notions related to interoperability — "functional interchangeability" suits the case of your comment. (See also the related item C1.2, and related change E40.)	E22
18	C3.7	When it comes to applications; several do exist and are already used in practice. This for example includes the setting of secure storage and outsourcing of cryptocurrency signing keys. What is noticeable with this setting is that threshold key generation is not needed. Another concrete use-case, but where threshold key generation is essential, is the key management-in-the-cloud solution as for example realized by companies such as Sepior or Unbound. In this setting the goal is to distribute the security when storing and constructing highly sensitive keys from a HSM, to distinct servers at distinct locations running distinct software. Finally, threshold schemes can also be relevant as components in other schemes. This is for example the case for Verifiable Delayed Functions (see for example https://eprint.iacr.org/2018/627), which can be based on RSA, but has the requirement that the factorization of the public key is unknown to all parties. Thus threshold RSA key generation is needed for parameter generation. VDFs are again very relevant in the blockchain/cryptocurrency universe.	– **NOTE:** The relevance of distributed key-generation varies with the application setting (e.g., see Section A.4). In some cases a key already exists as part of an application setup, and the goal is simply to prevent the key from continuing in a single location, which can for example be achieved based on secret-sharing, without generation of any other key. In some other settings it can be essential to ensure that the initial generation of a key (symmetric or public) is performed in a distributed fashion . – **CHANGED:** In Section A.4, added a sentence including the example related to RSA-based VDFs.	E60
19	C3.8	Thanks again for the great work. I am already a member of the google group related to this project. I hope you can use my comments and please don't hesitate to contact me if you would like me provide further details. I hope I can be of assistance throughout the standardization process.	– **RELATED:** C3.1 – **NOTE:** Thank you again for the encouragement and your valuable comments. As mentioned in the document, "Engaging with stakeholder is a priority in this project, toward an informed definition of criteria for standardization of threshold schemes for cryptographic primitives." (Section 1.1) and "NIST values the expert technical feedback from stakeholders and that feedback will be incorporated in the standardization process." (Section 7) – **CHANGED:** Added a note to the acknowledgments.	E8

#	Item id	**Comment set C4**: Svetla Nikova & Vincent Rijmen COSIC (KU Leuven, Belgium)	Notes and changes	Edit id
20	C4.1	We are organizing a workshop in the Lorentz Center, The Netherlands, April 28th until May 1st, where we will discuss with all the participants on the draft NISTIR 8214A, or a follow-up document, if available, and provide a common feedback document. In the meantime, here are already our thoughts.	– **NOTE:** Thank you for your engagement in seeking further common feedback from the community. – **CHANGED:** No change.	
21	C4.2	We agree with the proposed split-up between the multi-party setting and the single-device setting. Since we are interested mainly in the single-device setting, we will restrict our feedback to this setting.	– **RELATED:** C8.1 – **NOTE:** We have carefully considered as beneficial this split in two "domains". – **CHANGED:** No change.	
22	C4.3	**On benchmarking:** · We think that the following should be included among the evaluation criteria: circuit area, number of clock cycles, max frequency. Also the number of pseudo-random bits consumed should be taken into account. Alternatively, one could demand that all designs include the circuitry necessary to generate the required amount of pseudo-random bits.	– **NOTE:** The mentioned metrics can be important to assess efficiency and practical feasibility. – **CHANGED:** In Section 6.1, we mentioned these aspects as examples of detailed metrics to consider.	E50
23	C4.4	We believe that the existence of open-source reference implementations is an absolute requirement for an open benchmarking process.	– **RELATED:** C10.12 – **NOTE:** We see the value in open-source reference implementations. As mentioned in Section 6.1, we consider the existence of open-source reference implementations as a relevant aspect to consider for criteria. – **CHANGED:** No change.	
24	C4.5	We think that the benchmarking process may benefit from the definition of a small number of reference platforms (FPGA as well as ASIC), including measurement setups, for example the ChipWhisperer platform (FPGA). Alternatively, one could aim to develop a benchmarking framework similar to Athena (Automated Tool for Hardware Evaluation) or the SUPERCOP (benchmarking of cryptographic software).	– **NOTE:** A proper benchmarking analysis should enable fair comparison across different techniques. It can be useful to have reference platforms that enable such comparison across a wide range of techniques. – **CHANGED:** In Section 6.1, added a sentence highlighting this aspect.	E53
25	C4.6	The benchmarking process may benefit from the definition of a small number of reference functionalities (full AES, certain gadgets, . . .). **On useful gadgets.** We believe that it is good to concentrate on the AES as the final goal for secure implementations. Useful gadgets to reach that goal are: AND circuits, finite-field multipliers, AES S-box, AES round. On the other hand, new developments might result in demands for other gadgets, hence additional gadgets should not be excluded.	– **NOTE:** As mentioned in Section 4.2.1, AES is an item of focus. – **CHANGED:** In Section 5.3.2, added two examples (finite-field multipliers, AES S-box) in the discussion of gadgets.	E45

#	Item id	**Comment set C4**: Svetla Nikova & Vincent Rijmen COSIC (KU Leuven, Belgium)	Notes and changes	Edit id
26	C4.7	On threat models. We believe that the following threat models make sense: 1. Passive adversary that can collect 100k, 1 million, 10 million or 100 million traces.	– **RELATED:** C4.8, C9.3 – **NOTE:** It is important to quantify/estimate/reference the capabilities of conceivable adversaries. – **CHANGED:** In Section 4.2.1, added a simple sentence alluding to the relevance of considering how many traces an adversary may be able to collect.	E31
27	C4.8	2. Active adversary that can inject a small number (1 to 4) of controlled faults (controlled value and controlled location) and collects 100k, 1 million, 10 million or 100 million traces. It is important here to specify the size of the fault (1 bit, 1 byte, ...) and to specify if one counts the number of faults during the full execution or rather the number of faulted locations. (This makes a difference for implementations that re-use circuit components.) Typically, this adversary is assumed to be non-adaptive during a single AES execution. 3. Active adversary that can injects faults with random values on a small number of controlled locations (1 to 4) and collects 100k, 1 million, 10 million or 100 million traces, cf. [3].	– **RELATED:** C4.7, C9.3 – **NOTE:** It is important to quantify/estimate/reference the capabilities of conceivable adversaries. – **CHANGED:** In Section 4.2.2, added a sentence alluding to various capabilities of an active adversary.	E34
28	C4.9	In each of these cases, one may distinguish between (partial) key-recovery attacks and simple detections of leakage.	– **NOTE:** It is important to characterize what is the adversarial goal of an attack. – **CHANGED:** In Section 4.2.1, added half a sentence mentioning the example of a key recovery attack.	E32
29	C4.10	As reference documents for the definition of some desirable properties, we propose the following papers: [1] Begüül Bilgin, Benedikt Gierlichs, Svetla Nikova, Ventzislav Nikov, Vincent Rijmen: Trade-Offs for Threshold Implementations Illustrated on AES. IEEE Trans. on CAD of Integrated Circuits and Systems 34(7): 1188-1200 (2015) [2] Svetla Nikova, Vincent Rijmen, Martin Schlääffer: Secure Hardware Implementation of Nonlinear Functions in the Presence of Glitches. J. Cryptology 24(2): 292-321 (2011) [3] Oscar Reparaz, Lauren De Meyer, Begül Bilgin, Victor Arribas, Svetla Nikova, Ventzislav Nikov, Nigel P. Smart: CAPA: The Spirit of Beaver Against Physical Attacks. CRYPTO (1) 2018, LNCS 10991: 121-151 [4] Lauren De Meyer, Victor Arribas, Svetla Nikova, Ventzislav Nikov, Vincent Rijmen: M&M: Masks and Macs against Physical Attacks. IACR Trans. Cryptogr. Hardw. Embed. Syst. 2019(1): 25-50 (2019)	– **NOTE:** Thank you for pointing out these references. – **CHANGED:** No change.	

#	Item id	**Comment set C5**: Chelsea Komlo (University of Waterloo)	Notes and changes	Edit id
30	C5.1	In our recent work on real-world use cases of Threshold Schemes [1], we discuss the variance among security properties that are currently provided by different implementations of threshold schemes. Clearly defining security properties of general threshold schemes and how or if they differ among each design and use case during the standardization process would be helpful. [1] https://petsymposium.org/2020/files/papers/issue2/popets-2020-0033.pdf	– **RELATED:** C1.2, C2.3, C3.6, C7.4 – **NOTE:** We find important that threshold schemes specified for upcoming standardization have well defined security properties, including with considerations about their use cases. Thank you for pointing out this reference. – **CHANGED:** No change.	
31	C5.2	Another property to consider is forward secrecy of secret shares, such that any participant can coordinate a "ratcheting forward" of shares among the group without exposing the secret. We discuss a mechanism to perform this in [1], Section 9.	– **NOTE:** It is important to achieve advanced security properties, such as those that prevent a complete breakdown even when some assumptions are broken. – **CHANGED:** In Section 5.2.2, added a mention to forward secrecy.	E38
32	C5.3	Shamir Secret Sharing is already post-quantum secure (because it is information-theoretically secure). However, if the channel over which shares are distributed is not post-quantum secure (such as TLS today), then the scheme is no longer post-quantum secure. As such, assessing post-quantum security and where gaps arise during composition and real-world use could be helpful.	– **RELATED:** C1.2, C3.6 – **NOTE:** We are interested in assessing the security properties of threshold schemes proposed for standardization, namely in comparison with the properties of the original schemes. It is important to consider the effects of instantiating a communication protocol in a distributed computation protocol. – **CHANGED:** Added a bullet item "cryptographic assumptions" in Section 5.2.3.	E40
33	C5.4	Rogaway, in a recent talk at Real World Crypto, presented a formalization of threshold schemes, and compared threshold schemes to symmetric encryption schemes. I thought this was a good frame of reference to understand intended security properties and general use cases. For example, formalizing the step of generation and validation of an integrity value during share generation and distribution is important for secret sharing schemes, as is also the case in symmetric encryption schemes during encryption and decryption.	– **NOTE:** Thank you for pointing out this reference. It is important to have the analysis of threshold schemes supported on well defined security properties. – **CHANGED:** No change.	

#	Item id	**Comment set C6**: Nigel Smart (COSIC – KU Leiven)	Notes and changes	Edit id
34	C6.1	The comment about EdDSA might be hard to make threshold variant as it is deterministic is true for the non-hashed version. For the HashEdDSA it is not so true. We have a paper on this which we will post to ePrint soon; and we have already shared with NIST.	– **NOTE:** Among the versions of EdDSA considered in FIPS 186-5 (draft), the pre-hashed version (HashEdDSA) requires less distributed hashing in the case of a threshold implemnentation. However, it remains that the need for distributed hashing (using currently approved hash functions in the SHA2 or SHA3 families) "creates a technical difficulty" in the sense of being significantly costlier than a probabilistic version of Schnorr signatures. – **CHANGED:** In Section 4.1.1.3, edited some text to clarify the meaning of technical difficulty (inefficiency), even for HashEdDSA.	E28
35	C6.2	If you replace the hash function by an MPC friendly one such as Rescue then EdDSA or HashEdDSA are very simple and efficient in a threshold variant. Again will post to ePrint soon on this.	– **RELATED:** C2.5 – **NOTE:** See C2.5. – **CHANGED:** Added text reflecting on how to possibly incorporate building blocks into larger standards.	E46

#	Item id	**Comment set C7**: Phillip Hallam-Baker (Venture Cryptography)	Notes and changes	Edit id
36	C7.1	Threshold cryptography represents one of the greatest untapped resources in computer security. While practical threshold techniques have been known to the field since the 1990s, practical implementations remain few and far between. This paper is an attempt to bring together my experience of building a large cryptographic system that makes extensive use of threshold techniques and to make recommendations for future NIST work as requested in the solicitation [1]. A sketch of a proposed threshold co-processor for ECDH operations is provided in an appendix. [1] L. T. A. N. Brandão, M. Davidson and A. Vassilev , "Towards NIST Standards for Threshold Schemes for Cryptographic Primitives: A Preliminary Roadmap," NIST, Michael Davidson (NIST), Apostol Vassilev (NIST), 2019.	– **NOTE:** Thank you for the comments. – **CHANGED:** Added a note to the acknowledgments.	E8
37	C7.2	By far the most important recommendation is that NIST should focus on specifying threshold modes for the ECDH algorithms already in use. While this does limit the range of threshold capabilities available, the limitations are very small and have not proved significant in the context of building systems.	– **NOTE:** We have a current focus on NIST-approved primitives. This includes various primitives based on discrete-log and integer-factorization related assumptions, as well as others. Key-agreement based on Diffie-Hellman type assumptions is considered in SP 800-56A, and so it is a possible "primitive" to consider. – **CHANGED:** Added item Pair-Wise Key-Establishment Schemes Using ECC in Section 4.1.1.5.	E14, E15, E30

#	Item id	**Comment set C7**: Phillip Hallam-Baker (Venture Cryptography)	Notes and changes	Edit id
38	C7.3	THRESHOLD CRYPTOGRAPHY IN THE MESH The Mathematical Mesh (Mesh) [2] is a personal PKI designed with the goal of making computers easier to use by making them more secure. The Signal protocol [3] and system demonstrated that a messaging service offering end-to-end confidentiality could be as easy to use as an insecure one provided that certain design constraints were imposed. The Mesh extends this approach to support all the major forms of Internet messaging including in groups such as on mailing lists and in social media with transparent 'zero-impact' security. The need to support these capabilities naturally led to the use of threshold decryption techniques. This in turn led to the application of threshold techniques in other aspects of the design. The work presented is currently self-funded. The Mesh reference library is released under an MIT License and is not (to my knowledge) covered by any unexpired patent claims. [No warranty for this is given. It is the responsibility of users, implementers etc. to determine that they have the necessary licenses etc.] [2] P. Hallam-Baker, "Mathematical Mesh 3.0 Part I: Architecture Guide," 16 1 2020. [Online]. Available: https://www.ietf.org/id/draft-hallambaker-mesh-architecture-12.html. [3] Signal, "Signal," [Online]. Available: https://signal.org/. [Accessed 10 2 2020].	**– NOTE:** We expect that threshold cryptography can be useful for many applications. **– CHANGED:** No change.	

#	Item id	**Comment set C7**: Phillip Hallam-Baker (Venture Cryptography)	Notes and changes	Edit id
39	C7.4	UNGROUNDED SECURITY REQUIREMENTS CONSIDERED HARMFUL The problem of using encryption to control distribution of confidential documents was solved in the early 1990s. Yet use of 'CRM' systems remains the rare exception. Current commercial products are based on proprietary protocols that effectively end at the Enterprise boundary and employ single-point-of-failure key servers that do not provide end-to-end security. Controlling distribution is straightforward, controlling onward re-distribution is not. The re-distribution problem can only be addressed through the use of trustworthy computing platforms, a constraint which inevitably reduces reach to a tiny portion of the Enterprise space.* While most desktop and laptop computers now ship with an onboard Trusted Platform Module (TPM), a quarter century after work began, none of the platforms come close to supporting the application level capabilities required to address re-distribution. [* This market failure may be understood by noting that much of the IPR required to implement CRM was originally developed with DRM (i.e. copyright enforcement in mind). The acquisition of strong DRM IPR portfolios by the copyright stakeholders allowed them to dictate the direction of CRM product development. As a result, modern PCs have TPM modules with features suited to copyright enforcement requirements.] The insistence on a solution to the re-distribution problem cost the community the opportunity to deploy a solution to the simpler distribution problem. Yet, a solution that does not control redistribution may offer a better choice when considering the full systems requirements. Consider the case in which Alice Bob and Carol join a chat room that allows exchange of pictures and conversations with end-to-end security. Percy is admitted to the group and starts sending Child Sexual Abuse Material (CSAM). If the system allows Percy to control re-distribution, he can prevent Alice, Bob and Carol reporting the CSAM to the authorities as they would wish. Nor can providers of chat clients provide the option of scanning inbound materials for CSAM using resources such as PhotoDNA [4]. [4] Microsoft, "PhotoDNA," [Online]. Available: https://www.microsoft.com/en-us/photodna. [Accessed 10 2 2020].	– **RELATED:** C3.6, C5.1 – **NOTE:** It is important to assess what are the pertinent security requirements for a threshold scheme, including with respect to envisioned deployment/application contexts. For example, there are various challenges in communication, some of which rely on cryptographic primitives. Since we are focused on threshold schemes for cryptographic primitives, the definitional security properties of the original scheme are of interest. Naturally, it is also important to be considerate of the security implications of thresholdizing a cryptographic primitive. – **CHANGED:** No change.	
40	C7.5	A similar issue arose early in the development of public key cryptography. It was generally assumed that the Diffie Hellman key exchange does not fully solve the requirements of a 'true' public key encryption system since it does not provide for message recovery. But from the system designer's point of view, message recovery was never an issue as the public key system is only used to establish session keys used to encrypt the body of the message.	– **RELATED:** C7.2 – **NOTE:** For the purpose of thresholdization, we consider key-agreement and decryption as distinct primitives. – **CHANGED:** No change.	

#	Item id	**Comment set C7**: Phillip Hallam-Baker (Venture Cryptography)	Notes and changes	Edit id
41	C7.6	One of the recurring objections made to the key co-generation scheme used in the Mesh is that it is implicit that the key contributions be passed between the party by means of a secure channel, but the specification of the algorithm does not specify how, this communication is described as 'out of band'. This is entirely intentional and from the system designer's point of view, the correct choice because the choice of 'out of band' mechanism used will depend entirely on the characteristics of the implementation. In the real-life circumstances for which the Mesh is designed, establishing a sufficiently secure out of band channel is frequently practical. When onboarding an IoT device, at least one of the following out of band channels is usually available in a form that is sufficiently secure for the purpose: • A wired channel (USB, ethernet). • Short range wireless communications (near field, Bluetooth, WiFi). • QR code displayed on administration device. If we wish to make absolutely sure (or as certain as we can be) that a device being onboarded has not been compromised in any manner whatsoever, we are going to require the use of tools such as Trusted Facilities and Ceremony. Such controls are of course routine in some contexts but hardly appropriate for the task of onboarding a garage door opener to my smart-home hub.	– **NOTE:** It is important to consider the environment in which threshold schemes may be deployed, and what security properties may be required for their secure operation. In particular, it is relevant to take into account the properties of the communication medium. – **CHANGED:** No change.	
42	C7.7	To understand where the boundaries between cryptographic components lie, it is necessary to build systems that solve actual problems.	– **NOTE:** It is important to consider the boundaries of cryptographic components, namely in the scope of threshold systems. – **CHANGED:** No change.	

#	Item id	**Comment set C7**: Phillip Hallam-Baker (Venture Cryptography)	Notes and changes	Edit id
43	C7.8	BRING YOUR OWN KEY CONSIDERED HARMFUL Besides being compatible with deployed code, one of the chief differences between the Ed25519/Ed448 Signature scheme I have proposed to the IRTF CFRG and the BLS scheme currently being considered by the group is that it does not support a capability I call 'Bring Your Own Key' (BYOK). The omission is intentional and flows in part from the earlier realization that attempts to support 'true' proxy re-encryption are unnecessary. In a BYOK threshold scheme, Alice Bob and Carol all generate their public key pairs and then decide to perform some threshold cryptography together as a group. In a 'true' proxy re-encryption scheme Alice can encrypt a message and send it to a service that can then determine how to generate copies of the message encrypted under the keys of Bob and Carol without decrypting the message or being capable of decrypting it. Starting with the assumption that I needed a proxy re-encryption scheme, applying many successive rounds of simplification proved that I could eliminate almost all the complexity of the system and offer tight, compelling security proofs by making the group administrator responsible for generating all user keys used to decrypt messages sent to the group. The idea of Alice generating keys for Bob and Carol might appear an anathema but why is this? Alice is the administrator of the group; it belongs to Alice. Alice generates the original encryption keypair for the group and continues to hold the private decryption key. Having the keys that would be used by Bob or Carol to perform their part of a threshold decryption does not affect the security of the system in any way.	– **NOTE:** Proxy re-encryption may be useful in various applications. BYOK systems may be useful in some applications. – **CHANGED:** No change.	
44	C7.9	Allowing users to bring their own keys provides them with the ability to defect in imaginative ways and that inevitably requires greater complexity in the design. Greater complexity at the algorithmic level is only justified if it allows reduction in complexity or increased functionality at the systems level. Generating ECDH key pairs is cheap. BYOK makes no sense in the case of threshold encryption schemes. In the context of Threshold Signatures, BYOK does offer a (modest) functional benefit: A very modest saving of space. If Alice, Bob and Carol are using the 'Ecocide' cryptocurrency and wish to add transactions to its blockchain, use of a threshold signature allows the use of one signature instead of three. While this may look compelling if we only consider the space used by the signature data ($3\times100 = 300$ bytes becomes 100), it is rather less so when we consider the total block size ($3\times550 = 1650$ bytes becomes 1450). The limited use made of threshold signatures in the Mesh has not required BYOK. The fact that the approach proposed for the Mesh does not meet the needs of crypto-currency speculators is considered a feature rather than a bug.	– **NOTE:** For each considered threshold scheme, it is important to consider the complexity in design and algorithms, as well as at the system level. Saving of space and performance is an important consideration for application use-cases of threshold schemes, including for cryptocurrencies. Depending on the application, a 3-fold reduction in a signature size may signify a valuable gain, e.g., of 3-fold, in some metric of interest. There may be additional benefits in enabling aggregation of signatures, e.g., of more than 3 parties, and in avoiding reliance on a trusted dealer. – **CHANGED:** No change.	

#	Item id	**Comment set C7**: Phillip Hallam-Baker (Venture Cryptography)	Notes and changes	Edit id
45	C7.10	THRESHOLD CRYPTOGRAPHY IN THE MESH The Mesh was not developed as a showcase for threshold cryptography, but threshold techniques have since pervaded every aspect of the design. Use of Threshold cryptography permits a different, simpler approach to device onboarding, the provision of capabilities to devices and new end-to-end secure communication patterns. The Mesh uses a highly restricted palette of cryptographic algorithms, all of which are industry standards: • AES-256 • SHA-2-512, SHA-3-512 (including HMAC modes.) • Ed448 Signature • X448 Key agreement	– **RELATED:** C3.3 – **NOTE:** It may be advantageous, when possible, to use standardized primitives when building larger systems. – **CHANGED:** No change.	
46	C7.11	As a general rule, a threshold scheme that requires the use of different public key algorithms is of little or no interest in developing the Mesh. Persuading the field to think about the use of threshold techniques at all is challenge enough.	– **RELATED:** C1.2, C2.3, C3.6, C5.1, C5.3, C7.10 – **NOTE:** The matter of which primitives enable a threshold scheme is relevant, namely in comparison with the corresponding traditional (non-threshold) cryptographic primitive. – **CHANGED:** No change.	
47	C7.12	It was found that in almost all the cases in which threshold techniques are used, the number of shares n is the same as the threshold t and in most cases $n = t = 2$.	– **NOTE:** Threshold parameters may vary with the application and security requirements. In some cases a system intends to tolerate the failure of certain components, and achieves that property as long as the compromise threshold does not surpass a certain proportion of the total number of components. We mentioned the need to consider threshold parameter in Section 5.2.1. – **CHANGED:** No change.	
48	C7.13	The use of Shamir Secret Sharing and LaGrange coefficients does offer potential advantages in enforcing separation of duties in administration and fault tolerant provision of host services. These will be researched further after the first phase of the Mesh is completed.	– **RELATED:** C7.19 – **NOTE:** Shamir Secret sharing if a useful technique underlying various potential threshold schemes. We mentioned secret sharing as a gadget in Section 5.3.1. – **CHANGED:** In Section 5.3.1, briefly mentioned separation of duties.	E44

#	Item id	**Comment set C7**: Phillip Hallam-Baker (Venture Cryptography)	Notes and changes	Edit id
49	C7.14	THRESHOLD ALGORITHMS To allow detailed examination of the cryptographic techniques in isolation and to encourage the reuse of the approaches in other contexts, the threshold techniques used in the Mesh have been separated from the main Mesh document and submitted to the IRTF Crypto Forum Research Group. The base draft [5] describes: • The principles that enable the use of unanimous threshold ($n = t$) techniques in ECDH algorithms. • The use of Shamir Secret Sharing and LaGrange coefficients to extend this to the case $t < n$. • Adjustments to the X25519, X448, Ed25519, and Ed448 specifications to support threshold use • Key Co-Generation for X25519, X448, Ed25519, and Ed448. • Threshold decryption with X25519 and X448. The companion draft [6] describes: • Threshold signature with Ed25519, and Ed448. The precise extent of the handling of threshold signatures is not yet decided. [5] P. Hallam-Baker, "Threshold Key Generation and Decryption in Ed25519 and Ed448," 5 1 2020. [Online]. Available: https://www.ietf.org/id/draft-hallambaker-threshold-00.html. [6] P. Hallam-Baker, "Threshold Signatures Using Ed25519 and Ed448," 5 1 2020. [Online]. Available: https://www.ietf.org/id/draft-hallambaker-threshold-sigs-00.html.	– **NOTE:** Is is useful to allow detailed examination of proposed cryptographic techniques. – **CHANGED:** No change.	
50	C7.15	KEY CO-GENERATION Weak key generation is a major cause of cryptographic failures. In the IoT space, in addition to the recognized risk of weak keys being generated through use of insufficiently random, seed generation techniques, the risk of compromise in the supply chain is high. When originally proposed, Pedersen’s Distributed Key Generation (DKG) was conceived as a means of generating threshold key shares in a distributed fashion rather than generating a key pair in the usual fashion and splitting it. Co-operative key generation applies the same principles but with the goal of generating a single key pair as the output rather than a collection of threshold key shares. Co-operative key generation is used to create device keypairs for a device being onboarded ’Onboard’, using an administration device ’Admin’. 1. Onboard generates a key pair d.P, d and transmits the value d.P to Admin. 2. Admin generates an activation seed r with sufficient (i.e. 256 bits or more) strength. 3. Admin derives the activation key a.P, a from r using a specified deterministic KDF. 4. Admin calculates the activated device public key (d+a).P = d.P + a.P and creates relevant credentials for the key (d+a).P. 5. Admin passes the activation seed r OUT OF BAND to Onboard encrypted under the key d.P. via a secure channel*. [* The secure channel currently used for this in the Mesh applies a further layer of encryption at the presentation and transport layers. Thus the activation seed is encrypted three times.] 6. Onboard receives and decrypts the value r 7. Onboard derives the activation key a.P, a. 8. Onboard calculates the activated device private key d+a.	– **NOTE:** Generation of high-quality keys is an important aspect of building secure system, including in the threshold context, as mentioned for example in Section 4.1.1.4 and Section 4.2.1.2. – **CHANGED:** No change.	

#	Item id	**Comment set C7**: Phillip Hallam-Baker (Venture Cryptography)	Notes and changes	Edit id
51	C7.16	This approach addresses multiple security objectives: • Provision a device credential (d+a).P • Prevent leakage of the device identity through leakage of the onboarded device key. • Provide a degree of protection against weak or compromised onboarded device keys. • Provide a degree of protection against weak or compromised administration device keys. This approach does not provide perfect security in the case that a device is shipped with weak or compromised keys. But it does significantly restrict the window of vulnerability which is all that is reasonably achievable in the circumstances. If the adversary knows all the internal state of a device, can predict any random numbers generated by the device and can observe all communications between the device and the outside world, they can emulate the device and recover any information it contains. But the fact that a security control is ineffective against an omnipotent adversary should not discourage us from shutting the door on the less omnipotent.	**– NOTE:** There are numerous security properties of interest with respect to protecting secret keys. They should be measured in the context of adversarial settings with adversaries with goals and capabilities. **– CHANGED:** No change.	
52	C7.17	GROUP ENCRYPTION One of the chief limitations in the encryption capabilities provided by OpenPGP and S/MIME is the lack of support for end-to-end encrypted mailing lists. While both existing standards allow an encrypted message to be sent to multiple recipients, the full list of recipients must be known by the sender at the time the message is sent. This restriction has obvious limitations in an enterprise setting where membership of task groups changes dynamically over time. When Alice transfers her responsibilities for the W project, Bob needs full access to all the past discussions within the group. This includes documents, video conferences and chat logs in addition to all the mail messages exchanged. The Mesh uses a form of threshold encryption (n = t = 2) in which the group administrator creates a master X-448 encryption keypair from which unique threshold share pairs are created for each member added to the group. One of the threshold shares is sent to the member and the other is sent to a decryption service. [* Both threshold shares are transmitted under multiple layers of encryption with the innermost layer being a dedicated key for managing group subscriptions.] This separation allows the decryption service to control decryption of the documents without being able to decrypt them. If a member should leave the group, the decryption service is told to refuse further decryption requests from that member. The decryption service serves as a policy enforcement point for the access control system allowing sophisticated access criteria to be implemented (e.g. metered use) and provide accounting capabilities. The same approach has since been applied to mediate device access to a user's Mesh account data. This allows a user to immediately disable access to their password catalog and similar sensitive data if a device is lost or stolen.	**– NOTE:** Threshold encryption can be used in multiple application contexts. It is important to be aware of single points of failure. **– CHANGED:** No change.	

#	Item id	**Comment set C7**: Phillip Hallam-Baker (Venture Cryptography)	Notes and changes	Edit id
53	C7.18	FAULT TOLERANCE OF NOTARY SERVICES Threshold signatures are used in the Mesh for the very limited purpose of allowing the notary services to be made fault tolerant without introducing the risk of signing inconsistent notarized outputs or exposing the internal structure of the notary service to others. A system in which P, Q and R all sign the notarized value $n(t)$ might end up in a condition in which P and Q have signed $n(t)$ and E has signed $n'(t)$. While this risk can be mitigated through measures such as mutual verification, etc., the use of threshold signatures allows it to be eliminated. To establish the notary service online signature key, the administration device generates a set of key parameters (Shamir Secret Sharing). These are then used to generate key shares for P Q and R with a threshold of 2. The key shares are then distributed to the services. This provides a threshold signature scheme which precisely matches the intended fault tolerance criteria.	– **NOTE:** Threshold signatures have multiple potential application. Fault-tolerance Cna be a useful property – **CHANGED:** In Section A.5, added a mention to notary services, and mentioned the possible desirable feature of hiding the structure of the signers.	E61
54	C7.19	SEPARATION OF DUTIES Separation of duties is an important security control in an enterprise environment. The current Mesh design does not provide for separation of duties, but it is easy to see how $t < n$ threshold techniques could be employed to achieve this within the current framework.	– **RELATED:** C7.13 – **NOTE:** Separation of duties can be a useful feature in diverse processes. – **CHANGED:** In Section 5.3.1, included a very brief sentence mentioning the use-case of supporting separation of duties.	E44
55	C7.20	RECOMMENDATIONS BUILD THRESHOLD STANDARDS ON EXISTING STANDARDS Threshold cryptography has much to offer the field. But there is no time to wait for the field to become comfortable with a whole new generation of cryptographic algorithms before making use of it. If threshold cryptography is to succeed it must be seen as a set of operational modes for existing algorithms and not as a discontinuity. A large part of the power of the Mesh in use is that the public key operations associated with threshold keys are identical to those for single private key cryptography. Threshold encryption is merely regular X-448 encryption. Threshold signature verification is simply Ed-448 signature verification. While there may well be additional advantages to be gained from use of cryptographic techniques such as pairing, we must not let the perfect be the enemy of the good. There are plenty of benefits to be realized from defining threshold modes for existing ECDH algorithms and that should be the first priority before any new algorithms are considered.	– **RELATED:** C1.5, C3.3 – **NOTE:** With respect to continuity with existing standards of cryptographic primitives, in Section 6.4 we mention the possibility that some threshold schemes may appear as addenda to existing standards, conceivably as new “operational modes” of said primitives. The choice of which schemes to focus on should also consider the complexity of the constructions themselves. – **CHANGED:** In Section 6.4, added a small note one the intention of new stadards.	E57

#	Item id	**Comment set C7**: Phillip Hallam-Baker (Venture Cryptography)	Notes and changes	Edit id
56	C7.21	DEFINE A CRYPTO CO-PROCESSOR STANDARD Side channel attacks continue to be the chief cause of key disclosure. The chief side channels leading to disclosure today being disposal of storage media still containing the private key, backing up the private key to offline storage and (increasingly) uploading the signature key to GitHub. Trustworthy Computing has to date been an abject failure from the point of view of the computer user. And this is not surprising when the entire architecture of the TPM devices has been designed to serve only the copyright enforcement stakeholders. Trustworthy computing could have been used to bind TLS server keys to the hosts on which they were used. But the only benefit the user has got from these technologies is the ability to use full disk encryption using Bitlocker and luks. Threshold cryptography provides a new way to leverage trustworthy hardware at a time when it is desperately needed. MELTDOWN, SPECTRE and ROWHAMMER have demonstrated the vulnerability of modern multi-core CPUs to new forms of side channel attacks. While work is underway to mitigate these attacks, a limited ECDH coprocessor could eliminate them while providing a performance benefit.	– **RELATED:** C8.2, C7.24 – **NOTE:** Side-channels attacks are a major concern for which threshold schemes have some mitigation potential. – **CHANGED:** No change.	
57	C7.22	APPENDIX A: ECDH CO-PROCESSOR STRAW-MAN The following is proposed as a strawman proposal for a minimal ECDH co-processor. The chief objective of this proposal is to demonstrate that such a co-processor need not require an excessive number of gates or non-volatile memory. MASTER SEED. The device is initialized with a master seed 'MasterSeed' during manufacture. This is used to generate secret scalars for use with particular curves as follows: SecretScalar = KDF (MasterSeed, CurveID, BootID, l) Where KDF is as specified in RFC5869 (or equivalent). SecretScalar is the secret scalar for the curve named CurveID in the boot context BootID. CurveID is an identifier specifying the curve and mode of operation (key exchange, signature). BootID is an optional parameter being a cryptographic digest of the root of trust used by the boot sequence. A co-processor MAY permit the user to cause the MasterSeed to be reset to a random value. Though this function would prevent the attestation of the use of the co-processor to third parties.	– **RELATED:** C7.2 – **NOTE:** See reply to C7.2, about the pertinence of ECDH. – **CHANGED:** No change.	
58	C7.23	SUPPORTED CURVES The recent decision of the IETF/IRTF to standardize on the 'CFRG' curves MUST be respected and these MUST be supported at minimum. The NIST curve p-384 has achieved widespread use in the WebPKI and thus SHOULD be supported. The need for support for p-521 is unclear. It is unfortunate that the preferred curves are taken from different families. The CFRG specifies Montgomery curves for key exchange and Edwards curves for Signature. Meanwhile NIST specifies Weierstrass curves for both. Fortunately, it is possible to convert between representations and an ECDH co-processor could thus be implemented for a single curve form with translation between forms being performed in a device driver outside the trust boundary. For our straw-man we select the following curves: • Ed-25519 • Ed-448 • p-384 The choice of curve may be made according to whatever requires the fewest number of gates.	– **NOTE:** The NIST specifies "Recommendations for Discrete Logarithm-Based Cryptography: Elliptic Curve Domain Parameters" in SP 800-186. – **CHANGED:** In Section 4.1.1.4, added mention to the NIST document that considers recommendations for elliptic curve parameters.	E29

#	Item id	**Comment set C7**: Phillip Hallam-Baker (Venture Cryptography)	Notes and changes	Edit id
59	C7.24	SIDE CHANNEL RESISTANCE Co-processors SHOULD make use of Kocher side channel resistance techniques. To obtain the scalar product c.P, the processor first generates a random number x. The processor then generates x.P and (c-x).P and adds the results. This approach ensures that the processor does not perform repeated point multiplication operations on the value c, thus defeating statistical side channel approaches.	**– NOTE:** It is important to consider various techniques for mitigation of side-channel attacks. **– CHANGED:** In 4.2.1.3 included a mention to side-channel resistance enhanced by techniques that prevent the repeated use of the secret as input to some side-channel sensitive operation.	E33
60	C7.25	SUPPORTED OPERATIONS Three operations are supported: **Multiply P → c.P** The point P is multiplied by the secret scalar c. **Generate Blind _ → r.P** The register r is filled with a random value strictly smaller than the group prime and the value r.P returned. **Blind Multiply and Erase k → r + c.k** The value (r + c.k) is returned and the register r is cleared. An error is returned in the case that this instruction is attempted without first performing Generate Blind. These operations are sufficient to support threshold signature and key exchange algorithms compatible with traditional elliptic curve operations without disclosing the secret scalar value as follows.	**– NOTE:** It is important to identify sets of basic operations that may be used to support one or various threshold schemes for one or various cryptographic primitives. **– CHANGED:** In Section 5.3.1, mentioned that it can be useful to identify basic operations that support threshold schemes.	E43
61	C7.26	KEY GENERATION Let the SecretScalar value for the desired operation on the specified curve be c. To make use of the co-processor, the application generates a secret scalar a in the normal fashion. To generate the application's composite public key, the application calculates $a.B+c.B$ where B is the base point. The value $c.B$ is obtained using the Multiply co-processor operation. KEY AGREEMENT Key agreement is performed in precisely the same way as key generation except that it is the point P provided by the counterparty that is multiplied by the a and s scalars: The user calculates $a.P+c.P$ where P provided by the counterparty. The value $c.P$ is obtained using the Multiply co-processor operation	**– NOTE:** Key generation and key agreement are primitives of interest in the scope of threshold schemes, as mentioned for example in Section 4.1.1.4. **– CHANGED:** No change.	

#	Item id	**Comment set C7**: Phillip Hallam-Baker (Venture Cryptography)	Notes and changes	Edit id
62	C7.27	SIGNATURE To perform a threshold signature operation using a Schnorr signature scheme such as Ed25519/Ed448 requires two rounds of processing. Recall that a Schnorr signature has the form (z, R) where z is given by: $z = r + (k * s) \mod Q$ where $k = H(R, m)$, m is the message to be signed, Q is the group order. Converting the single key Schnorr signature to a threshold scheme requires that each participant protect their secret scalar value from all the others by specifying a unique blinding factor r. Moreover, it is essential that a blinding factor is never re-used under any circumstance. It is therefore necessary to ensure that once a blinding factor is created, it is only used once. To minimize the need to lock the co-processor for an extended time we perform as many of the preparatory steps as possible prior to requesting the co-processor perform a Generate Blind instruction to minimize the time until the Blind Multiply and Erase operation is performed. 1. 1. Application generates random blinding factor r_a, calculates $R_a = r_a.B$ 2. 2. Application obtains the value Rc from the co-processor using the Generate Blind instruction 3. 3. Application calculates $R_s = R_a + R_c$ 4. 4. Application calculates $k = H(R_s, m)$ 5. 5. Application obtains the value $z_c = (r_c + c.k) \mod Q$. from the co-processor using the Blind Multiply and Erase instruction 6. 6. Application calculates the value $z_a = (r_a + a.k) \mod Q$. 7. 7. Application calculates the value $z_s = z_a + z_c$. The signature value is (z_s, R_s) The chief weakness of this approach is that a given co-processor can only perform one threshold signature operation at a time. While this is likely to be more than sufficient for cases in which the signature is being produced by a single device, it might prove inconvenient for networked applications in which a threshold signature is being generated by multiple independent hosts. One approach that might prove useful is the use of a counter mode in which the random blinding factor r_c is generated from a KDF and a monotonically increasing counter. In this approach, the co-processor need only use 1024 bits of state plus the state required to encode the last unused counter value to track 1024 outstanding blinding factor commitments.	**– NOTE:** It is important to consider how high-quality randomness is obtained for use in threshold schemes. **– CHANGED:** Mentioned randomness as one basic operation for which it is useful to consider how to generate in a setting of side-channel attacks.	E43

#	Item id	**Comment set C8**: Ventzi Nikov (NXP Semiconductors)	Notes and changes	Edit id
63	C8.1	Some thoughts on draft NISTIR 8214A for the single-device setting since I am only interested in this setting. Below I express my personal opinion and it may not reflect the view of my employer NXP Semiconductors.	– **RELATED:** C4.2 – **NOTE:** Thank you for your comments – **CHANGED:** No change. Added a note to the acknowledgments.	E8
64	C8.2	We well know that modeling side channel leakage as wire probing and fault effects again as probing gives too much power to the attacker. In reality the attacker gets by far less info. Still we use them in theoretical modeling because they suit best our current understanding how to design countermeasures and more importantly how to prove/argue about their security. There are other models some of which are closer to how hardware leaks, for example bounded-moment leakage model, however there is no theory yet of how to argue security in such models.	– **RELATED:** C2.3, C7.21, C8.2, C7.24 – **NOTE:** It is important to be aware of realistic models, as well as models that enable formal reasoning about security. – **CHANGED:** No change.	
65	C8.3	Building on top of the wire probing model one can impose necessary conditions for a design to have no leaks or impose even stronger but sufficient conditions. One can implement securely a cipher satisfying only the necessary conditions but not the sufficient ones. Moreover only few gadgets are known which satisfy the necessary and sufficient conditions and their cost is higher compare to gadgets which satisfy only the necessary conditions. By cost I mean not only the silicon area, but cost can be a power or energy budget, or latency, etc.	– **NOTE:** It is useful that implementation guidance makes clear what is necessary and what it sufficient to achieve a certain security validation profile. – **CHANGED:** No change.	
66	C8.4	Secure implementations must satisfy variety of (product) requirements for both cost and security. Since these requirements may differ a lot I believe the industry need is to have flexible tools to achieve different levels of security and at the same time be cost-efficient. Flexibility implies to have a variety of techniques or gadgets which one can use to achieve his goals. In my opinion all techniques must obey the necessary conditions but only the highest security levels can eventually satisfy the sufficient conditions as well.	– **RELATED:** C1.5, C9.5 – **NOTE:** Modularity is an important feature, as discussed in Section 5.3. Having well-defined gadgets and well considered composition rules may enable some useful flexibility in threshold schemes. – **CHANGED:** In Section 5.3, added text emphasizing the importance of modularity and flexibility in a context of continuous innovation.	E42
67	C8.5	I think the standard has to focus on order of security which do resonate in practice for example up to d=8, but focus more on the small orders e.g. up to d=4. From practical point of view nobody is interesting to know an asymptotic (on d) cost of a gadget. If at some future moment attackers become so good that they can break the designs with d=8 then we better look for other ways to protect implementations (e.g. on protocol level or requiring more often re-keying), since otherwise the cost of protection will be unacceptably high.	– **RELATED:** C4.3, C9.3 – **NOTE:** The practicality of some of these parameters may become more clear based on benchmarking. Re-keying is a complementary approach that can be considered nonetheless, e.g., in the sense of pro-active rejuvenation (Section 5.2.2). – **CHANGED:** In Section 6.1, added some notes on benchmarking in the scope of the single-device setting.	E50, E51

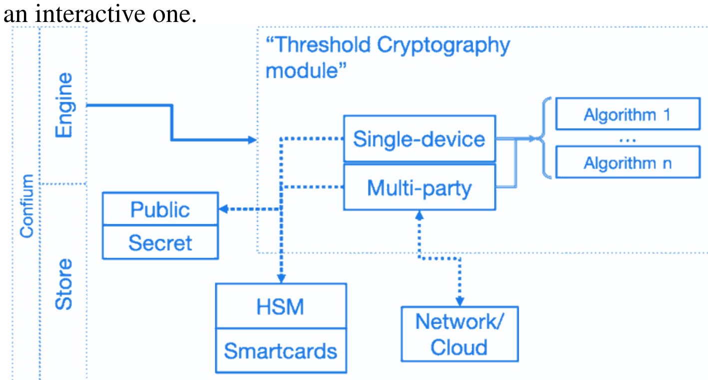
#	Item id	**Comment set C9**: Simon Hoerder & Elke De Mulder (Rambus Inc.)	Notes and changes	Edit id
68	C9.1	We welcome NIST's efforts to consider side-channel attacks and countermeasures for standardization. Based on our extensive experience in developing secure IP cores and past experience from Inside Secure/Rambus in developing smart card products we would like to propose areas for special attention to NIST:	– **NOTE:** – **NOTE:** Thank you for the encouragement and comments. – **CHANGED:** Added a note to the acknowledgments.	E8
69	C9.2	Provide guidance for a holistic approach to secure system design. Justification: Not all system integrators manufacturers are equally capable in security topics and a tick-box exercise that requires expensive side channel countermeasures for a system with a complex software stack and corresponding risk surface. Avoiding remotely exploitable software vulnerabilities needs to be the first priority. Similarly, any scenario where side-channel attacks are a valid attack scenario, fault-injection attacks must be considered as well.	– **NOTE:** Upcoming standards should be helpful with respect to secure system design and deployment, including with respect to conceivable side-channel and fault-injection attacks. – **CHANGED:** In Section 6.4, added a noteemphasizing the intended utility of upcoming standards.	E58
70	C9.3	Limit the life time of keys within a holistic system design. Justification: Having provable security against 1st, 2nd or 3rd order attacks is nice but meaningless if an adversary can easily obtain a nearly infinite number of traces for higher order attacks. Also, the number of traces needed to pick up leakage of a specific order is not always increasing at higher orders: In our experience, 1st order leakage will always need less traces than 2nd order leakage and 2nd order leakage will always require less traces than higher order leakage but we have seen, for example, a case where 5th order leakage was easier to detect than 3rd order leakage. In order to avoid having to test for such intricacies it is best practice system engineering to e.g. limit the life time of keys such that even in the worst case only a finite number of traces (e.g. somewhere between 100k and 1M traces) can be obtained per key. However, this requires strict life cycle management and adequate infrastructure support.	– **RELATED:** C4.7, C4.8, C8.5 – **NOTE:** Some benchmarking may help understand how (in)feasible it is for an attacker to achieve certain rates of collection of traces. Rejuvenation is an important feature for threshold schemes. – **CHANGED:** In Section 6.1, added a mention to benchmarking the feasibility of rates of collection of traces.	E51
71	C9.4	Beyond guidance for system design, focus on test and certification procedures that integrate with hardware and software development processes. Justification: Any improvement in test and certification procedures will help to provide clarity about the security levels achieved by competing products and thus will strengthen the market for secure products by providing a level playing field. Integration of test and certification procedures into all steps of the development process will help to shorten time-to-market for new products and improve market agility as it will help to identify potential issues as early as possible in the design process when they are still relatively easy to rectify.	– **NOTE:** It is important that certification corresponds to clarify of security properties achieved by an implementation. – **CHANGED:** In Section 5.1, added one sentence mentioning the suggested integration.	E36

#	Item id	**Comment set C9**: Simon Hoerder & Elke De Mulder (Rambus Inc.)	Notes and changes	Edit id
72	C9.5	Standardizing one or more particular countermeasures must only be done in such a way that it doesn't disadvantage alternative designs that achieve similar security assurances. Justification: Today we have a healthy and diverse market for DPA protected products. Market participants have optimized their countermeasure stacks based on required security strength, available technologies and patent concerns to meet customer requirements for different market segments such as payments, content protection and national security. A misguided standardization of a particular countermeasure risks skewing or even destroying this market without providing a clear gain for the general public.	– **RELATED:** C1.5, C8.4 – **NOTE:** ... – **CHANGED:** In Section 5.3, added text emphasizing the importance of modularity and flexibility in a context of continuous innovation.	E42

#	Item id	**Comment set C10**: Ronald Tse, Wai Kit Wong, Daniel Wyatt, Nickolay Olshevsky, Jeffrey Lau (Ribose Inc.)	Notes and changes	Edit id
73	C10.1	Please find attached our submission of comments in response to the public comment period of Draft NISTIR 8214A. We hope they are useful for the NIST threshold cryptography standardization efforts.	– **NOTE:** Thank you for comments. – **CHANGED:** Added a note to the acknowledgments.	E8
74	C10.2	Comments in response to public comment period of Draft NISTIR 8214A ["Ribose Whitepaper 11021:2020"] **Contents.** 1. Scope. 2. Normative references. 3. Terms and definitions. 4. Practical usage of new cryptographic families (4.1. Background; 4.2. Key management; 4.3. Threshold cryptography; 4.4. Adoption challenges; 4.5. Architecture concerns). 5. Decoupling cryptographic primitives in trust stores (5.1. Challenges; 5.2. Drivers for an extensible architecture; 5.3. Requirements for a trust store). 6. The Confium trust store (6.1. Purpose; 6.2. Background; 6.3. Structure; 6.4. Security analysis). 7. Crypto-primitive layer (7.1. Modular extension of cryptographic schemes; 7.2. Plugin types; 7.3. Identification and organization of cryptographic schemes; 7.4. Third-party modules; 7.5. Threshold cryptography; 7.6. Security requirements). 8. Keystore layer (8.1. General; 8.2. Private keystore; 8.3. Public keystore; 8.4. Plugins; 8.5. Access control; 8.6. Security requirements). 9. Public module repository (9.1. General; 9.2. Security requirements). 10. Confium offers support to the NIST threshold cryptography project. 11. Confium feedback to NIST 8214A (11.1. Threshold cryptography benefits to OpenPGP; 11.2. Alignment to NISTIR 8214A Figure 2 cryptographic modes). 12. Supplementary information (12.1. Development approach of Confium; 12.2. Information about Ribose). Bibliography	– **NOTE:** The submitted white paper is too large to include in these notes, so in the left cell we are copying the index of the document. In the next items we focus on sections 4.3, 7.4, 10 and 11, which are more closely related to threshold cryptography and NISTIR 8214A. – **CHANGED:** No change.	

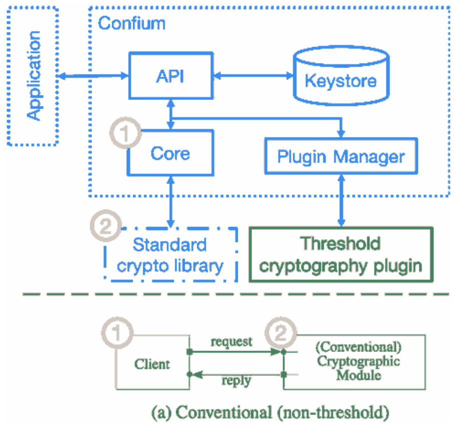
#	Item id	**Comment set C10**: Ronald Tse, Wai Kit Wong, Daniel Wyatt, Nickolay Olshevsky, Jeffrey Lau (Ribose Inc.)	Notes and changes	Edit id
75	C10.3	**4.3. Threshold cryptography.** A simple application that demonstrates the value of threshold cryptography is multi-factor authentication. Suppose there are three items to prove the identity of a user: a) user password; b) a one-time password (OTP) from the user's phone; and c) the user's fingerprint. A user needs to provide any two of the above items to login the system. This can be done using threshold cryptography with $t = 2$ and $n = 3$. The advantages of such scheme are that: a) even if one item, say the user password, is stolen by an attacker, the attacker cannot login the system; and b) when a user loses one item, say the user forgets his password, the user can still be authenticated and an authenticated reset of the key(s) can be done.	– **NOTE:** Threshold schemes do provide tolerance to compromises, such as availability (in case a user forgets a password) and confidentiality (in case an attacker stoles a password). The case $(t, n) = (2, 3)$ has been exemplified in NISTIR 8214. – **CHANGED:** No change.	
76	C10.4	These properties of threshold cryptography can clearly improve overall security and ease-of-use for the user. While many threshold cryptography algorithms [13] [10] [26] [14] have been developed, they all differ in the algorithms and/or security models to achieve thresholdization. There is no standardized interface of threshold cryptography today, and NIST is currently attempting to standardize algorithms in this field. Without a common standardized interface, it is difficult for an application to make use of threshold cryptography: – If an application selects to implement a particular algorithm of threshold cryptography, it may not fit the user's needs that require another implementation of threshold cryptography with some other features. – It is ineffective for the application to implement multiple algorithms for the user to choose. Only with a standardized interface, we can separate the application logic from cryptographic tools. Applications can be developed according to the definition of the interface, and any implementation that conforms to the standardized interface can be selected by the user and run smoothly with the application. [13] C. Delerable é and D. Pointcheval. Dynamic threshold public-key encryption. In Crypto, 2008. [10] D. Boneh, X. Boyen, and S. Halevi. Chosen ciphertext secure public key threshold encryption without random oracles. In CT-RSA, 2006. [26] X. Yan, Y. Lu, L. Liu, S. Wan, W. Ding, and H. Liu. Chinese remainder theorem-based secret image sharing for (k, n) threshold. In ICCCS, 2017. [14] C. Hazay, G. L. Mikkelsen, T. Rabin, T. Toft, and A. A. Nicolosi. Efficient RSA key generation and threshold paillier in the two-party setting. J. Cryptology, 32(2), 2019.	– **NOTE:** Section 2.3 discusses some I/O interfaces from the client-perspective. There is interest in being aware of application use-cases, so that upcoming standards are applicable. – **CHANGED:** No change. Added one sentence highlighting the "keep it simple" aspect on the side of the client.	E17
77	C10.5	**4.4. Adoption challenges. 4.4.1. Overview.** Threshold cryptography introduces challenges that are not currently addressed by traditional cryptographic libraries and implementations. The two major areas include the implementation and adoption of cryptographic schemes in real-world applications, and interoperable facilitation of secrets utilized by those schemes.	– **NOTE:** As mentioned in Section 1.2.2, it is important to focus on where there is a high need and high potential for adoption. – **CHANGED:** No change.	

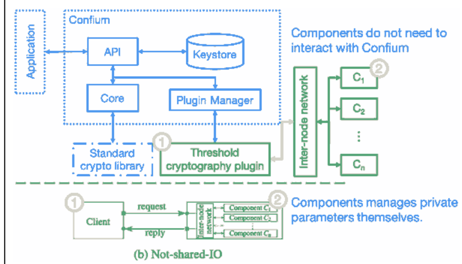
#	Item id	**Comment set C10**: Ronald Tse, Wai Kit Wong, Daniel Wyatt, Nickolay Olshevsky, Jeffrey Lau (Ribose Inc.)	Notes and changes	Edit id
78	C10.6	**4.4.2. Implementation and availability.** An application may use a number of cryptographic schemes in order to achieve different functionality. For an application to utilize a new cryptographic scheme, short of directly implementing the scheme within the application, generally requires all the following conditions to be met: a) The cryptographic scheme has been standardized by one or more standardization development bodies, such as NIST, ISO, ITU and IETF. The standardization timeline has a multi-year horizon and is largely driven by national agenda. b) The (standardized) cryptographic scheme has been adopted by one or more standard cryptographic libraries, such as OpenSSL, LibreSSL, mbtls, BoringSSL. At this stage stakeholders of the scheme will have to contribute and implement this scheme for the major cryptographic libraries, each with different requirements, security implications and timelines. This stage is often a multi-year process. c) The standard cryptographic library that implements the cryptographic scheme has been adopted by an operating system vendor or distributor, such as Apple, Microsoft, IBM and Oracle. There is a typical lag between step 2 and 3 of at least a year or more. These steps are mostly sequential — success in step 1 leads to step 2, etc. They lead to an adoption timeline, even in an optimistic sense if the cryptographic scheme and its use cases are fortunate enough to garner long-term and widespread support, of at least 5 years to over a decade. All of the above factors leading to the success of a cryptographic scheme are not directly linked to its purpose or technical merit, and are often completely out of the control of the creator and researchers of a cryptographic scheme. Herein lies the difficulty in incorporating threshold cryptography in real applications.	**– NOTE:** The time for adoption is a relevant matter. The process of standardization should require that motivating applications be identified, and reference implementations be made available. the timeline for testing and improving them should start before the publication of standards. **– CHANGED:** No change.	
79	C10.7	EXAMPLE. [22] developed a threshold signature scheme as an extension of the traditional RSA signature scheme, such that the signature is generated by multiple parties instead of one party. The method of signature verification is identical to traditional RSA, meaning that the verifier only needs to understand traditional RSA without the need to implement the scheme described in [22]. Applications that can practically adopt such threshold signature scheme are likely to be a decade out, even when the its mechanisms are based on existing, widely available, cryptographic primitives. [22] V. Shoup. Practical threshold signatures. In EUROCRYPT, 2000.	**– NOTE:** The motivation for using threshold schemes is multi-faceted. The standardization of threshold schemes should be based on compelling usefulness, including for the enhancement of best practices. **– CHANGED:** No change.	

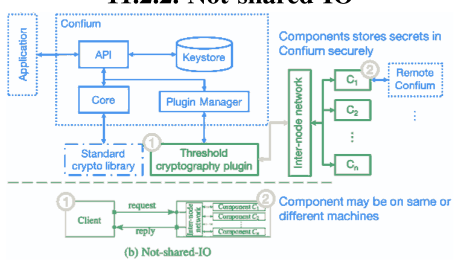
<table>
<tr><th>#</th><th>Item id</th><th>Comment set C10: Ronald Tse, Wai Kit Wong, Daniel Wyatt, Nickolay Olshevsky, Jeffrey Lau (Ribose Inc.)</th><th>Notes and changes</th><th>Edit id</th></tr>
<tr><td>80</td><td>C10.8</td><td>4.4.3. Secret storage management. Keystores are essential in the operation of encryption and signature schemes as they rely on the protection of secrets. Every primitive may define new types of secrets with different key lengths, properties and operations.
Traditional keystores, such as Oracle JKS1, assume that individual cryptographic schemes are independent, and thus each cryptographic scheme is implemented as a separate module without being extensible.
In addition, traditional keystores also rarely provide an interoperable way for others to obtain the generated public keys. The user application, and often the user of the application, has to resort to out-of-band mechanisms to obtain the public keys of others in order to import them into one's own keystore. This process is opaque to users of the applications, and may introduce more security issues (such as improper sharing of secrets) compared to the enhanced security provided by the adoption of such cryptographic scheme.
In the realm of threshold cryptography, many cryptographic schemes are extensions of some others, and the integration between them and traditional keystores will be clumsy at best.
Moreover, today's keystores often rely on proprietary secret protection, leading to unwanted lock-in in the storage or keys, reduced resilience in face of application failure, ultimately increasing security risks of the organization. With the advent of multiple threshold cryptography schemes, an open, interoperable keystore will be necessary to manage the various types of secrets.</td><td>– NOTE: Interoperability is an important feature that can be facilitated by the development of a standards.
– CHANGED: Section C11.10 has enhanced the discussion about interoperability.</td><td>C11.10</td></tr>
<tr><td>81</td><td>C10.9</td><td>7.5. Threshold cryptography. Confium aims to support new cryptographic families and threshold cryptography is one of them. The threshold cryptography module is implemented in Confium's crypto-primitive layer, where threshold algorithms could: a) depend on existing cryptographic algorithms for calculations, such as threshold RSA to RSA b) have access its own private keystore, and the public keystore of the Confium keystore layer c) have access to hardware modules exposed by Confium d) access network interfaces if the scheme is an interactive one.

</td><td>– NOTE: It is important to consider how threshold cryptographic schemes may be implemented within products that rely on cryptographic operations.
– CHANGED: No change.</td><td></td></tr>
</table>

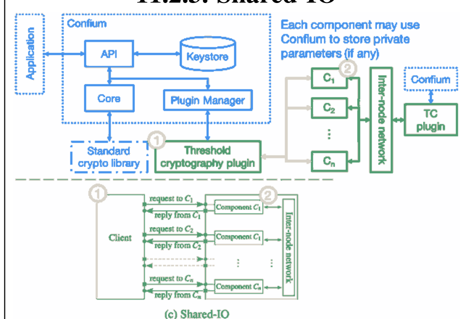
#	Item id	**Comment set C10**: Ronald Tse, Wai Kit Wong, Daniel Wyatt, Nickolay Olshevsky, Jeffrey Lau (Ribose Inc.)	Notes and changes	Edit id
82	C10.10	**10. Confium offers support to the NIST threshold cryptography project.** As developers of Confium, we strongly commend and fully support NIST's effort in the standardization of threshold cryptography.	– **NOTE:** Thank you for the encouragement. – **CHANGED:** No change.	
83	C10.11	We believe that the goals of Confium fully support and align with the current standardization efforts, especially in the areas of: a) Providing a common platform for cryptographers to develop prototype to production algorithms and schemes; b) Providing basic infrastructure primitives commonly used in threshold cryptography (e.g. networking code); c) Allows cryptographic testing in a sandbox to real-world deployment; d) Makes assessment easier by providing a level-playing field.	– **NOTE:** Implementation and benchmarking in reference platforms is important in the process, as enumerated in Section 6.1 – **CHANGED:** No change.	
84	C10.12	We would like to contribute effort in providing Confium to NIST as an open-source implementation test-bed for threshold cryptographic schemes. Specifically, we are willing to work with NIST in ensuring that the test-bed meets the requirements set by NIST.	– **RELATED:** C4.4 – **NOTE:** Enabling open-source implementations is useful, as enumerated in Section 6.1. – **CHANGED:** No change.	
85	C10.13	As a gesture of commitment, Confium will implement a proof of concept to demonstrate the capabilities of the test-bed. It will be a 2-out-of-3 threshold RSA signature scheme, where the secret key is shared across 3 parties, and any pair of them is able to sign or decrypt, but without the secret key ever being recombined.	– **NOTE:** It is interesting to learn about implementations of threshold schemes. – **CHANGED:** No change.	
86	C10.14	**11. Confium feedback to NIST 8214A. 11.1. Threshold cryptography benefits to OpenPGP.** The following clauses refer to Appendix A of NISTIR 8214A (Draft). a) Threshold cryptography could help store OpenPGP secrets in multiple shares, allowing the private keys to be recoverable. ("A.2 Protection of secrets at rest") b) An OpenPGP identity key can be stored in multiple shares such that only when multiple factors are provided the key could be used ("A.3 Confidential communication") **EXAMPLE 1.** A user may want to keep the private key in 3 shares (iCloud keychain, computer, USB key), where all 3 must be present to utilize it for multiple-factor authentication. c) An OpenPGP identity key can be distributed across secure environments ("A.6 Distribution of trust across secure environments") **EXAMPLE 2.** The user could keep an identity key across 3 shares where 2 must be present. This would allow the user to recover the key even if the computer is lost, but the iCloud keychain and USB key are still present.	– **NOTE:** Secret-sharing of secret keys is an example related to threshold cryptography. A more general goal is to enable their use without reconstruction of the secret key. – **CHANGED:** No change.	

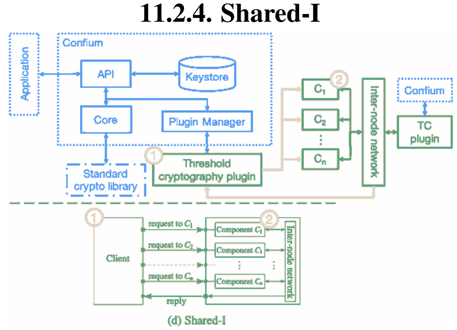
#	Item id	**Comment set C10**: Ronald Tse, Wai Kit Wong, Daniel Wyatt, Nickolay Olshevsky, Jeffrey Lau (Ribose Inc.)	Notes and changes	Edit id
87	C10.15	**11.2. Alignment to NISTIR 8214A Figure 2 cryptographic modes.** The following sections demonstrate how the Confium architecture aligns to the threshold cryptographic modes listed in NISTIR 8214A (Draft) Figure 2. In the following diagrams: a) The dotted box represents the scope of the Confium project. This includes an interface (API) with the user application. The application can request authentication to utilize the trust store's features. (does it have access rights to use this key in keystore to encrypt?) b) The Core cryptographic engine for handling standard encryption / signature using the standard cryptographic libraries. c) A plugin interface and a manager for managing and interacting with user-installed plugins (e.g., registration of new plugin)	– **RELATED:** C10.16 – **NOTE:** It is relevant to reflect on how threshold schemes may be implemented in connection with applications that use cryptographic primitives. – **CHANGED:** No change.	

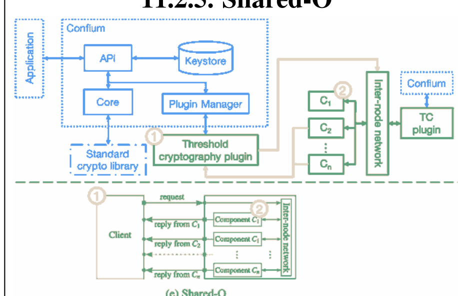
<table>
<tr><th>#</th><th>Item id</th><th>Comment set C10: Ronald Tse, Wai Kit Wong, Daniel Wyatt, Nickolay Olshevsky, Jeffrey Lau (Ribose Inc.)</th><th>Notes and changes</th><th>Edit id</th></tr>
<tr><td>88</td><td>C10.16</td><td>
11.2.1. Conventional


mode (a) alignment
11.2.2. Not-shared-IO


mode (b), alignment (i):
Component not storing keys in Confium
11.2.2. Not-shared-IO


mode (b), alignment (ii):
Component storing keys in Confium
11.2.3. Shared-IO


mode (c) alignment
11.2.4. Shared-I


mode (d) alignment
11.2.5. Shared-O


mode (e) alignment
</td><td>
– RELATED: C10.15
– NOTE: In some of the illustrated cases, the “Threshold Cryptography Plugin” seems to take the role of a “client” interacting with a threshold entity. It may be interesting to check whether this matches the use cases from a user perspective. Possible considerations: can the application avoid a single-point of failure with respect to its interaction having to be mediated by Confium? could a user (of the application) establish an extra direct connection with the TC plugin, to use with it key-shares not stored on the Confium “Keystore”?
– CHANGED: No change.
</td><td></td></tr>
</table>

#	Item id	**Comment set C10**: Ronald Tse, Wai Kit Wong, Daniel Wyatt, Nickolay Olshevsky, Jeffrey Lau (Ribose Inc.)	Notes and changes	Edit id

#	Item id	**Comment set C11**: Other editorial and content adjustments (from internal revision)	Notes and changes	Edit id
89	C11.1	Editorial adjustments in the covers and headers, related to switching from a *draft* to a *final* version.	– **CHANGED:** Remove "draft" from the NISTIR number: (E1); Remove draft from the doi: (E3); Update document date (E5).	E1; E3; E5.
90	C11.2	Adjust the title	– **CHANGED:** Updated the title to focus more on the "criteria";	E2.
91	C11.3	Update affiliation	– **CHANGED:** Updated one affiliation	E4.
92	C11.4	Other adjustments in the front-matter.	– **CHANGED:** Remove dates of public comment period.	E6
93	C11.5	Other adjustments in the front-matter.	– **CHANGED:** In the abstract, add clarifying note about "standards" not implying FIPS.	E7, E13
94	C11.6	Review the acknowledgments section	– **CHANGED:** Updated the Acknowledgments to mention the received public comments.	E8
95	C11.7	Patents disclosure notice	– **NOTE:** The ITL patent policy describes how to change the notice between the draft and the final version – **CHANGED:** Updated from "Call for Patent Claims" to "Patent disclosure notice"	E9
96	C11.8	Initial intuition about threshold property	– **NOTE:** Improve initial intuition about threshold schemes in the executive summary. – **CHANGED:** Improved the explanation of the threshold property, in both f-out-of-n and k-out-of-n perspectives, and added initial example about threshold RSA.	E10
97	C11.9	Initial intuition about secret sharing	– **NOTE:** Improve initial intuition about secret sharing – **CHANGED:** In Section 1, added a mention to secret-sharing as a base technique for threshold scheme, with some brief high-level explanation of its properties.	E11
98	C11.10	Interoperability-related notions	– **NOTE:** Need to better the relevance of interoperability and several of its nuances, namely those focusing on the client side and on a functional perspective. – **CHANGED:** Promoted the old subsubsection 2.3.3 (interchangeability) to a new subsection 2.4 (notions of interoperability for the client), with a better explanation and a clearer scope for notions like functional equivalence and interchangeability. To reduce confusion, these interoperability notions are now less associated with the qualifier "mode".	E19, E26

#	Item id	**Comment set C11**: Other editorial and content adjustments (from internal revision)	Notes and changes	Edit id
99	C11.11	Functional equivalence	– **NOTE:** It is useful to distinguish the restricted notion of functional equivalence, to better distinguish it from a broader notion of functional interchangeability. – **CHANGED:** In Section 2.4, defined functional equivalence equivalence and included two examples.	E20
100	C11.12	Functional interchangeability	– **NOTE:** Focus the scope of interchangeability more on the functional perspective, rather than on a deployment one. – **CHANGED:** Within the new subsection 2.4 on interoperability notions, included the definition of functional interchangeability as a concept broader than functional equivalence. Also included two examples. Briefly mention that the deployment perspective would be yet a different perspective. Add caution note for the need of a proper security analysis.	E21
101	C11.13	Interoperability-related notions	– **NOTE:** Clarify that ponderation is needed in the standardization process Mention also the need to concentrate efforts. – **CHANGED:** In Section 4, add initial paragraph discussing the need to ponder which items to standardize. Mention the usefulness of finding commonalities and synergies to ease the development.	E25
102	C11.14	Allowed use of keys	– **NOTE:** Make a more generic note about the allowed/disallowed use of keys. Sometimes the usage of keys should be restricted to certain types of operation, e.g., only signing, or only encryption/decryptions. – **CHANGED:** Remove small note from Section 4.1.1.2. Add a more elaborate note to Section E41.	E27, E21
103	C11.15	Phases of the process	– **NOTE:** In Section 6 of the draft version, phase 1 was about developing a roadmpa, and then there was a mismatch in phase 2 (the index mentioned “Calls” but the explanation mentioned “Develop Criteria”). – **CHANGED:** Phase 1 is now directly about developing criteria; phase 2 is about issuing calls for contributions; phase 3 is about evaluating contributions.	E47, E48, E54, E55
104	C11.16	Patents disclosure notice	– **NOTE:** Various editorial improvements of the text, e.g., reduce occurrences of “we”, clarify some paragraphs, ... – **CHANGED:** (Various changes across the document)	E16