NISTIR 8214A

NIST Roadmap Toward Criteria for Threshold Schemes for Cryptographic Primitives

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

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

NISTIR 8214A

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.*

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

July 2020

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
39 pages (July 2020)

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

Comments on this publication may be submitted to:

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

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 constitutes a preparation toward devising criteria for the standardization of threshold schemes for cryptographic primitives by the National Institute of Standards and Technology (NIST). The large diversity of possible threshold schemes, as identified in the NIST Internal Report (NISTIR) 8214, is structured along two main tracks: single-device and multi-party. 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 selection of items for standardization needs to consider diverse features, such as advanced security properties, configurability of parameters, testing and validation, modularity and composability (e.g., of gadgets vs. composites), and specification detail. Overall, the organization put forward serves as a preparation for an upcoming solicitation of feedback useful for considering a variety of threshold schemes, while differentiating standardization paths and timelines 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.

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.

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 thank Nicky Mouha and Matthew Watson for participating in discussions at numerous threshold cryptography meetings, Lily Chen and Andrew Regenscheid for additional feedback on 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.

Patent Disclosure Notice

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 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.

As of the date of publication and following call(s) for the identification of patent claims whose use may be required for compliance with the guidance or requirements of this publication, no such patent claims have been identified to ITL.

No representation is made or implied by ITL that licenses are not required to avoid patent infringement in the use of this publication.

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

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.

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. 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.

Threshold schemes can be applied to various cryptographic primitives, and in particular to NIST-approved algorithms. This includes the primitives that are part of asymmetric-key schemes, such as digital signatures (in FIPS 186) and key-establishment 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), and the related algorithms based on secret/private keys, such as signing, decryption within a public-key encryption (PKE) scheme, and enciphering and deciphering.

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 workshop (NTCW 2019) 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 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 the large diversity of possible threshold schemes, to identify priorities and to engage with the stakeholders for collaboration and feedback. To 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. Each domain can encompass various base cryptographic *primitives* and corresponding threshold *modes* of operation. The perceived difficulty of standardization varies with the items, namely based on 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 first includes the simpler cases and only later the more complex cases.

Not all conceivable threshold schemes are appropriate 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 complexities (e.g., building blocks vs. composites) and different levels of specification; the specification of advanced security properties (e.g., composability) required for secure deployment; the suitability for testing and validation guidelines, to address regulatory requirements; and the availability of configurability options (e.g., threshold values).

Using the outlined approach, this document identifies a diverse set of standardization objects (primitives and threshold modes) 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 under consideration may result in various distinct calls for contributions, as well as different timelines for the different 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, 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 depending on the corresponding complexity of the potential threshold schemes, and on criteria to be developed for their proposal, evaluation and selection.

The main purpose of this document is to 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 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.

Table of Contents

List of Figures

1 Introduction

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 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.

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 had already identified the need to devise criteria for eventual calls for contributions for the development of new standards of threshold cryptographic schemes. The present document (NISTIR 8214A) is intended to serve as a preparation for definition of criteria. The goal is to lay out reference rationale (complementary to what the 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.

The document covers various aspects pertinent to the standardization of threshold schemes for cryptographic primitives: 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 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 sophisticated algorithms and techniques that enable threshold schemes for some cryptographic primitives. 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 to generate feedback for the process ahead.

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. The threshold approach can mitigate the consequences of those attacks and make them costlier to execute. Therefore, standardizing threshold schemes may also contribute to new best security practices in cybersecurity.

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

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, appropriate parameters, and building blocks. Some of the latter are within consideration for standardization, but attaining 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.

This effort will inevitably lead to some open problems of interest to the research community. For example, threshold schemes are possible for candidate primitives under current evaluation within other NIST projects, such as the post-quantum cryptography and the lightweight cryptography. Although interesting, these cases are not considered 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. For some of the items identified in this document, a natural question is: *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.

In some cases, a reasonable end goal may be to add an addendum (e.g., of a simple threshold mode) in an existing standard; in others, an appropriate goal may be to devise reference definitions 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 that confirms certain threshold properties. Some cases may warrant specifying new algorithms. The concrete form in which to deliver the new standards will become apparent as the process moves forward.

A key note: 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

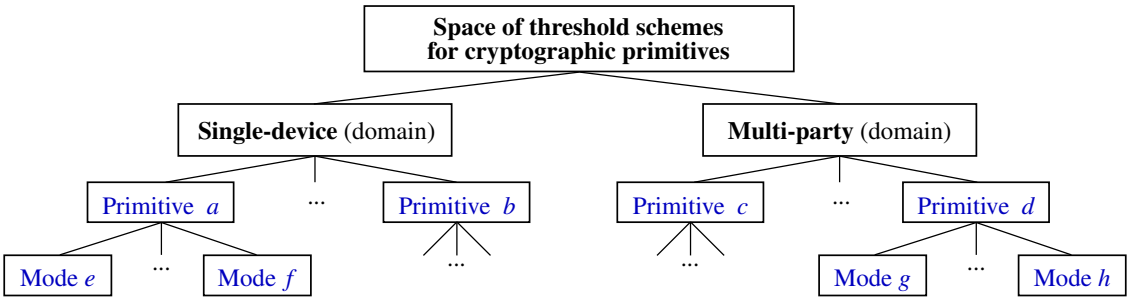

Figure 1. A depiction of a variety of primitives and threshold modes across two domains

1.2.1 The potential space of standardization

Considering several dimensions of the space of threshold schemes, the analysis of potential items for standardization benefits from a structured approach. It is useful to first distinguish between the single-device and the multi-party *domains* (Figure 1). Then, in each domain there are several cryptographic *primitives* 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 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, 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 increase resistance against attacks on implementations. 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 primitives. Along the process, stakeholders can provide feedback on concrete applications of interest.

1.2.3 Items across two tracks

At a high level, the standardization effort is organized into two separate **tracks** (one per *domain*): single-device and multi-party. The two domains differ substantially in system models, so the separation into tracks allows a better differentiation between concurrent approaches of standardization.

The organization of possible items (primitive/mode) for standardization can be independently organized per track. Some of the default 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 Standard (AES), and methods for random number generation (RNG). 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 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 identification of threshold modes of interest, there are detailed features of fundamental importance in the upcoming phase of criteria definition. Three important aspects are:

- *configurability and security features*, 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*, which is relevant to identifying recurring building blocks (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

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 developmental needs may be organized into different tailored calls for contributions and corresponding timelines. This improves 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, several phases present opportunities for public feedback. Currently, there is particular interest in the following topics:

1. standardization items (including threshold modes) that fit the described organization;
2. potential real-world applications that motivate concrete threshold schemes;
3. interface and security properties of interest in the threshold scope;
4. criteria for evaluating and comparing 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 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 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.

2 The space of threshold schemes for potential standardization

2.1 Two domains

As a way of organizing the potential space of standardization of threshold schemes, 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 tends to enable a modularized patching of components (e.g., repairing newly found bugs in existing components, or replacing old components 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 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. 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). 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; 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. 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, key generation (including RNG), and one-side key-operations related to pair-wise key-agreement.

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 current NIST-approved algorithms. One probable 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 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.

Several threshold techniques enable distributing — across several parties or components — the trust about the secrecy of a private key, such that the compromise of the internal state of a single party 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, 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. The overall design of those schemes must have well-understood security properties. 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 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 (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, 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, such as a ciphertext block $c = \text{AES}_K(p)$, 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, 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, a composite of components (either multiple parties, or a single-device with several components) enabling a threshold property for some security property.

Within the client–module paradigm, this document uses the notion of *threshold mode* to refer to the level of characterization 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.

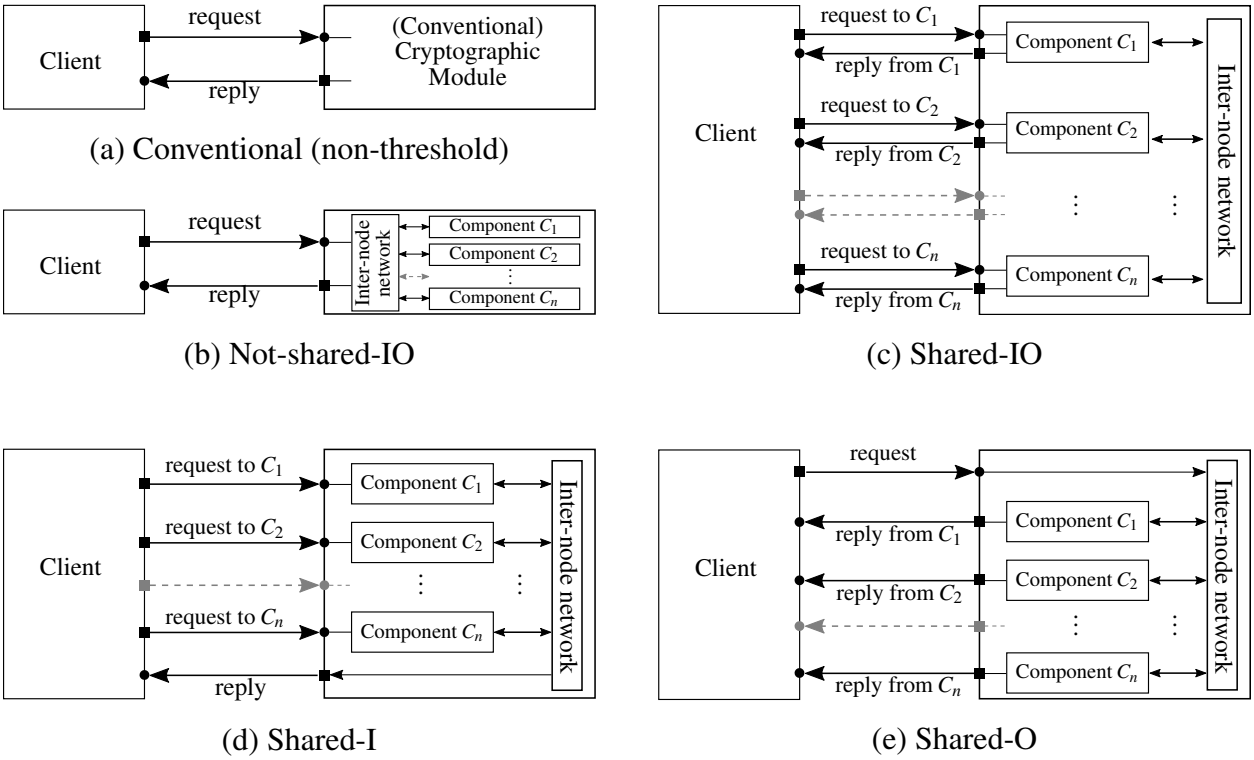

Figure 2. Several threshold interfaces (and one non-threshold case)

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/or on how to interpret the input and output. 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).

With respect to the input/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 input/output connections depend on the implementation and on a more detailed specification of the threshold scheme.

The following description conveys additional detail about the interfaces:

- **Not-shared-IO:** The client sends the full input to the threshold entity (via a relaying proxy or primary component, or by broadcasting to all components), and later receives back the output, exactly as in the non-threshold scheme. See Figure 2b.
- **Shared-I:** 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 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 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 client then reconstructs the final output from the shares. This mode is relevant for enhanced secrecy of output (e.g., a plaintext obtained from threshold decryption). See Figure 2e.
- **Shared-IO:** 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.

"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. The above distinctions apply well to primitives with a clearly defined input/output (e.g., ciphertext/plaintext). This includes cases where the needed secret or private key has 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 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 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 key.

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 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/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.

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 would interact in a 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. 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 is within the area of interest of the privacy-enhancing cryptography project.

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 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.

Functional equivalence. A threshold scheme for a cryptographic primitive is *functionally equivalent* to its non-threshold counterpart if for each input in the threshold scheme the output probability distribution is identical. Examples of functional equivalence:

- **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/output interface mode).
- **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 (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.
- **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 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 does not imply auditability. Even though a client uses secret-sharing, the client may remain unable to externally prove that a threshold computation took place. Conversely, a not-shared-IO mode may allow auditability, if the client receives complementary information (e.g., zero-knowledge proofs, or authenticated transcripts) that allow an external verification of the participation of multiple components with registered identities.

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.

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.

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. 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 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 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 various kinds. For example: of availability, by not requiring all parties to be present; 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, the focus is on applications in the active model, where corrupted parties can deviate arbitrarily from the protocol specification. As such, 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 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, and interacting hardware security modules.

4 Standardization 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.

This section describes at a high level some technical aspects required for threshold schemes for some primitives and modes being pondered for standardization. Since the two domains (multi-party and single-device) correspond to substantially different implementation scenarios, their corresponding standardization processes are referred to as different *tracks*. Furthermore, within each domain 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.

The initial emphasis is on obtaining threshold versions of NIST-approved conventional primitives. Some threshold schemes may 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.

Section 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.

The set of items identified ahead is not assumed to be exhaustive.

4.1 Multi-party track

4.1.1 Simpler cases

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, by a proxy, or (if it is not a problem to leak the clear plaintext) by the components of the threshold entity. The focus is on obtaining a not-shared-IO mode. There is also interest in considering the shared-I mode, 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.)

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 concern, for the

decryption operation it may also be relevant to consider the shared-O mode, i.e., as an enhanced way of preventing leakage of sensitive data.

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 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. The concrete (deterministic) EdDSA replaces the randomness by a hash of the concatenation of the secret signing key and the message being signed. 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 functionally equivalent threshold mode, compared to what would be possible for a probabilistic signature. This may either imply a more complex path of standardization, or additional possible considerations about the exact intended threshold mode.

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 operates on the public generator to produce the public key. Secret keys for the mentioned elliptic-curve signatures can be easily 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.

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.

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

4.1.2 More complex cases

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.

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

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 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 mainstream technology used at home and the office, people again turned to hardware for acceleration and security. For example, AES instructions and 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, 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. 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 better mitigation of side-channel leakage in hardware.

4.2.1 Simpler cases

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.

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.

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, and 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.

4.2.2 More complex cases

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). 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 is, 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.

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 (Section 5.1), configurability and security features (Section 5.2), and modularity (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 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 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 (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 (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 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 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)).

5.2 Configurability and security features

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

features may also depend on the considered application scenarios.

5.2.1 Threshold numbers

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. 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 with 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.

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 and/or have new requirements.

- **Composability.** 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 the adversary allowed to observe the protocol execution before deciding which components to corrupt?
- **Graceful degradation.** Is there a controlled vs. uncontrolled breakdown as soon as the threshold number of corruptions is surpassed?
- **Termination options.** How is the scheme characterized in terms of termination, e.g., with respect to fairness, guaranteed output delivery and identifiable abort?
- **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.

- **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 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.

5.3 Modularity

The 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, toward obtaining standardized threshold schemes that are concrete and provably secure instantiations of compositions of well-understood building blocks. The figure shows four symbolic quadrants, explained below.

5.3.1 Security definitions of building blocks (Q_1)

Reference definitions of abstract gadgets (e.g., 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 a gadget into a standard for the first time, the gadget should have a 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 of compiling from scratch arguments about the security of a more complex concrete protocol based on them, provided that composability properties are taken into consideration.

In a similar vein, it can be useful to identify the basic arithmetic operations (e.g., modular

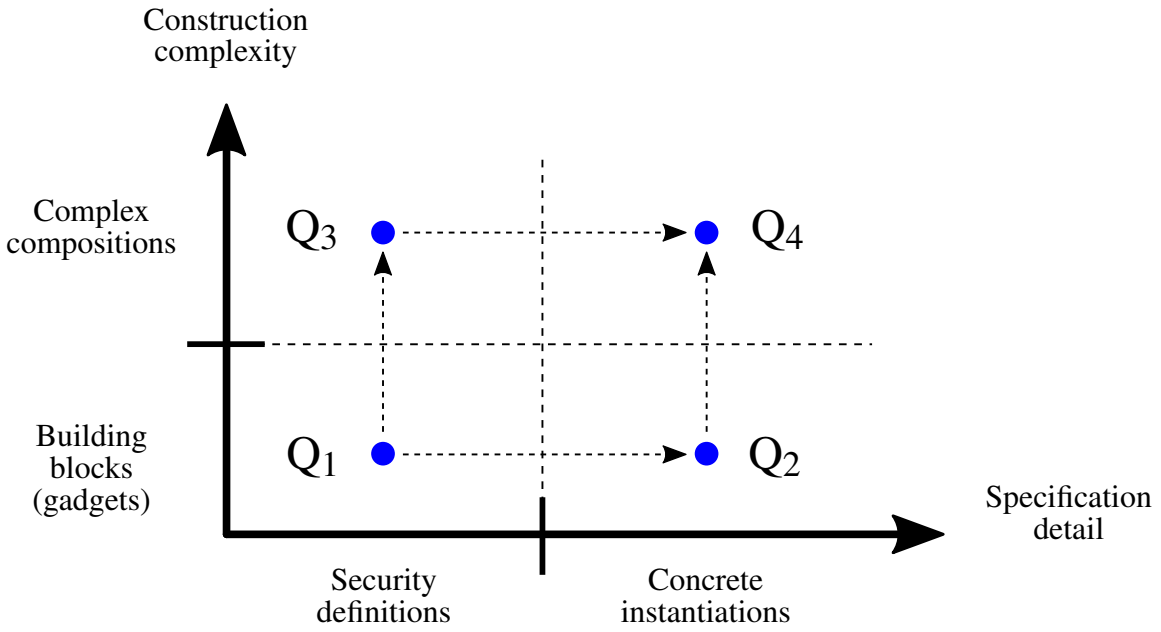

Figure 3. Modularity tradeoffs

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.

Secret sharing is a particular case of a gadget applicable across all primitives. 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. Standards for more complex threshold schemes 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.

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 compare concrete constructions of gadgets applicable across various threshold schemes. For example, for commitment schemes one 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 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.

5.3.3 Security definitions for complex compositions (Q_3)

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 protocols need to satisfy. 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, 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).

6 Development phases

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

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. **Criteria.** Develop differentiated criteria suitable for various foreseen standardization items.
2. **Calls.** Perform calls for contributions, based on criteria and timelines.
3. **Evaluation.** Obtain and evaluate input obtained in the context of calls for contributions.
4. **Publish.** Write and publish new standards and guidelines

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, possibly with distinct timelines. For some items, some phases may have several rounds, such as possibly alternating several calls for contributions (phase 2) and corresponding evaluations (phase 3).

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

a. Produce draft documentation and call for feedback.
b. Evaluate and integrate external feedback.
c. Publish documentation.

6.1 Phase 1 — Develop criteria

The NISTIR 8214 has already enumerated several representative questions to consider when reflecting on criteria. Some of the relevant aspects are:

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

The above items are important factors to take into consideration but are not themselves a specification of criteria. It is important to obtain further feedback about these items, and several of 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.

The goal of phase 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, 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 2 — Issue calls for contributions

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, different initial types of calls are envisioned (and described here at a high level):

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. These cases are not being envisioned as *competitions*, as it is more likely that different proposals share common features and it may be desirable to adapt features for some final protocols.

The more technically challenging items may require complex choices about their internal gadgets and their composition.

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.

It is a priority of this project to engage with the research community in 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 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 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 of the implementation of the corresponding cryptographic primitives, namely with respect to a wide range of side-channel and fault-injection attacks.

7 Collaboration with stakeholders

As an immediate 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.

NIST values the expert technical feedback from stakeholders and 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 process itself needs to lead the upcoming standards to be of high quality.

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

7.1 Multi-party setting

There is interest in the development of multi-party threshold schemes that improve key-confidentiality, as well as operational integrity and availability for the implementation of cryptographic primitives. 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

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

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.

[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.

Appendix A — Application use cases

This section describes at a high level several conceivable applications that take advantage of threshold schemes for cryptographic primitives. This is intended 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. Algorithms of asymmetric cryptography, such as RSA and ECC digital signatures, can be implemented by a hardware accelerator, 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. 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, and such implementations are also susceptible to fault attacks. 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 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 there is value in focusing on hardware implementations of block-cipher algorithms (AES strongly preferred) and 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 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 access to funds, in the case of a crypto-currency private key.

In any of the 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. 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 may allow the user to distribute shares of the key to several friends, such that if the 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, 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 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. 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, log their operations for future audits. Also, if the scheme is auditable 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. Thus, a KDC represents a single point of failure. If the KDC is offline, the clients cannot securely communicate or access needed services. 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, 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 secret share it across multiple parties of a subsequent threshold signing 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.

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 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 a functionally interchangeable mode, such that the output is syntactically equivalent to an original signature. 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 their own independent secret-public key pair. 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.

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.

For certain threshold schemes, such as 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; 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 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 as: (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 with a salt share, jointly compute the salted hash, but without 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 pseudo-random functions. These can be 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-sharing.