Draft NISTIR 8214A

Towards NIST Standards for Threshold Schemes for Cryptographic Primitives

A Preliminary Roadmap

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-draft

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

Draft NISTIR 8214A

Towards NIST Standards for Threshold Schemes for Cryptographic Primitives

A Preliminary Roadmap

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

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

November 2019

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 Report 8214A
36 pages (November 2019)

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

Public comment period: ***November 11, 2019,*** **through** ***February 10, 2020***

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 proposes a preliminary roadmap for the standardization of threshold schemes for cryptographic primitives by the National Institute of Standards and Technology (NIST). To cover the large diversity of possible threshold schemes, as identified in the NIST Internal Report (NISTIR) 8214, we tackle them in a structured way. We consider two main tracks — single-device and multi-party — and within each of them we consider cryptographic primitives in several possible threshold modes. The potential for real-world applications is taken as an important motivating factor differentiating the pertinence of each possible threshold scheme. Also, the standardization of threshold schemes needs to consider features such as configurability of parameters, advanced security properties, testing and validation, granularity (e.g., gadgets vs. composites) and specification detail. Overall, the organization put forward enables us to solicit feedback useful to consider a variety of threshold schemes, while at the same time considering differentiated standardization paths and timelines, namely depending on different levels of technical and standardization challenges. This approach paves the way for an effective engagement with the community of stakeholders and a preparation for devising criteria for standardization and subsequent calls for contributions.

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

Acknowledgments

This document follows the NIST Internal Report 8214 and the NIST Threshold Cryptography Workshop, which benefited from the participation and feedback from numerous individuals, representing various organizations. The authors of this document would also like to thank Nicky Mouha and Matthew Watson for participation in discussions at numerous threshold cryptography meetings, and Lily Chen and Andrew Regenscheid for additional feedback on this draft. We welcome public feedback until this document is finalized and published.

Call for Patent Claims

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

ITL may require from the patent holder, or a party authorized to make assurances on its behalf, in written or electronic form, either:

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

b) assurance that a license to such essential patent claim(s) will be made available to applicants desiring to utilize the license for the purpose of complying with the guidance or requirements in this ITL draft publication either:
   i) under reasonable terms and conditions that are demonstrably free of any unfair discrimination; or
   ii) without compensation and under reasonable terms and conditions that are demonstrably free of any unfair discrimination.

Such assurance shall indicate that the patent holder (or third party authorized to make assurances on its behalf) will include in any documents transferring ownership of patents subject to the assurance, provisions sufficient to ensure that the commitments in the assurance are binding on the transferee, and that the transferee will similarly include appropriate provisions in the event of future transfers with the goal of binding each successor-in-interest.

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

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

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

The most identifiable property of threshold schemes is that they enable essential security properties — such as secrecy of keys, integrity of computed values, and/or availability of operations — even when up to a certain threshold number of their components are compromised. Such schemes can be applied to various cryptographic primitives, and (for our purposes) particularly to NIST-approved algorithms, including those that are part of asymmetric-key schemes, such as digital signatures (in FIPS 186) and key-establishment (in SP 800-56A and SP 800-56B) based on integer-factorization cryptography (IFC) or on discrete logarithm cryptography (DLC), namely elliptic-curve cryptography (ECC), and symmetric-key schemes, such as block-cipher operations (in FIPS 197). The primitives of interest encompass key generation, including requirements related to random-bit generation (in SP 800-90 series), as well as the actual secret/private-key based algorithms, such as signing, decryption within a public-key encryption (PKE) scheme, and enciphering and deciphering.

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

The positive feedback received on the report (NISTIR 8214) and on the workshop (NTCW 2019) confirms 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 through the large diversity of possible threshold schemes, namely to organize, prioritize, and engage with the stakeholders for collaboration and feedback. To this effect, this document starts by orga-

nizing the standardization effort into two different domains: single-device and multi-party.

As confirmed by feedback in the workshop (NTCW 2019), these domains have significantly different challenges and involve different threshold considerations. Within each domain we can then consider various base cryptographic *primitives* and corresponding threshold *modes* of operation. Each item has their specific perceived difficulty of standardization, namely based on the existence vs. absence of related base standards and on the dependence on complex techniques. This makes it likely that future new standards are reached in a sequence that includes first the simpler cases and only later the more complex cases.

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

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

The end results of standardization may span new standalone documents as well as be incorporated as addenda (e.g., specifying threshold modes) in existing standards. Furthermore, different items of standardization can have different associated timelines, with the latter being shaped based on the corresponding complexity of the potential threshold schemes, namely with respect to criteria to be developed for their proposal, evaluation and selection.

The main purpose of this document is to solicit input for our roadmap to standardize threshold schemes for cryptographic primitives. This process includes for example obtaining technical comments about threshold schemes from experts in areas of threshold cryptography, strategic comments from those who work in cryptography standards but may be unfamiliar with threshold cryptography, and input about motivating application scenarios and restrictions from security practitioners and vendors.

Table of Contents

List of Figures

1 Introduction

NIST has established the Threshold Cryptography project to drive an effort to standardize threshold schemes for cryptographic primitives. Threshold schemes enable distribution of trust placed on human operators, and also offer a path to prevent several single-points of failure in conventional cryptographic implementations. This document comes on the heels of the NIST Internal Report (NISTIR) 8214, which posed representative questions about standardization of threshold schemes, and the NIST Threshold Cryptography Workshop (NTCW) 2019, which brought together a variety 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. This document (NISTIR 8214A) is intended to devise a preliminary roadmap for the standardization effort. A main motivation 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. This is still an early step that identifies at a high level the space of standardization, and a corresponding variety of manners to approach possible items, with possible different timelines.

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

1.1 A multifaceted standardization effort

Diverse stakeholders. The challenge inherent to this standardization endeavor goes beyond the technical considerations about the simple and the sophisticated algorithms and techniques that enable threshold schemes for some cryptographic primitives. We recognize 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 in this document is intended to engage all stakeholders and generate feedback about the roadmap 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, mitigating the consequences of those attacks and making them costlier to execute. Therefore, standardizing these 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, we consider several approaches, not all of which fall within the scope of developing new standards. For standardization, we are focused on threshold schemes for NIST-approved cryptographic primitives. We want to enable the standardization of threshold modes of implementation for these primitives, as a way to promote better best practices in settings where the use of these primitives is considered to be subject to adversarial attacks on the implementation or on the operation.

There are some simple to define threshold schemes applicable to some cryptographic primitives. There are also demonstrably feasible threshold schemes whose consideration still raises difficulties for the selection of the best techniques, and appropriate parameters and building blocks. For some of the latter we still aim for standards, but attaining them will require first establishing a clear rationale to support concrete selections.

This effort will inevitably lead to some open problems of interest to the research community. For example, threshold versions of candidate primitives under current evaluation within other NIST projects, such as the post-quantum cryptography and the lightweight cryptography, where the proposed conventional non-threshold primitives are still under security evaluation. Although interesting, these cases are not considered here as in scope for standardization. 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: *do we need a standard for this?* The question leaves implicit the meaning of *standard*, which may vary with the context. In some cases a reasonable end goal may be to add a simple addendum (e.g., of a simple threshold mode) in an existing standard; in others an appropriate goal may be to devise reference definitions (e.g., of secret sharing) that may appear as building block of several new techniques to consider; in some other cases a worthy goal may be to devise implementation guidelines that enable validation within a certain security profile level that confirms certain threshold properties; in some cases we may actually consider specifying particular new algorithms. The concrete form in which to deliver the new standards will become apparent as we move forward.

A key takeaway: we want to engage with stakeholders towards an informed definition of criteria for standardization of threshold schemes for cryptographic primitives.

1.2 A structured approach

1.2.1 The potential space of standardization

Since the space of threshold schemes has many dimensions, the analysis of potential items for standardization benefits from a structured approach. We start by distinguishing the single-device and multi-party *domains*. In each domain there is a potential applicability for several cryptographic *primitives*, and each of those can be potentially implemented 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,

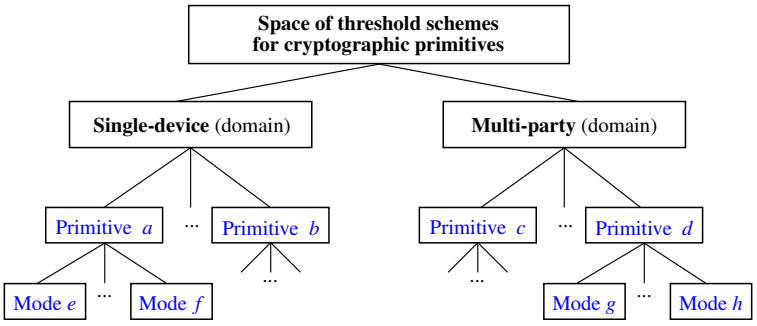

Figure 1. A depiction of a variety of primitives and threshold modes across two domains

a perceived difficulty need-not keep us away from advancing towards standardizing an item, even if it may take longer to achieve.

1.2.2 Motivating applications

While there are many conceivable threshold schemes, we consider important to focus on where there is a high need and high potential for adoption. An overarching motivation in this effort is developing the ability to distribute trust in operations, and increasing resistance against attacks on implementations, of 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. We can benefit in learning from stakeholders about more concrete applications.

1.2.3 Items across two tracks

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

For each track we are interested in organizing possible items (primitive/mode) for standardization. Some of the default potential primitives to 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 Advanced Encryption Standard (AES), and methods for random number generation (RNG). Within these, there is a special interest in the primitives related to secret keys, such as key-generation, signing, decryption within a public-key encryption (PKE) scheme, and symmetric-key enciphering and deciphering. For each primitive we are interested in considering what are the 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. This preliminary

roadmap emphasizes three aspects: *configurability and security features* — need to be specified in order 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); *modularity of components and specification detail* — relevant to identify recurring building blocks (such as secret sharing) that may appear across several threshold schemes, as well as improving the security analysis and the simplicity of specification.

1.2.5 Development phases

We intend to drive the standardization project in phases of devising criteria for calls for contributions, evaluating proposed contributions, and writing documentation for new standards. Standardization items with different development 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, the several phases present opportunities for public feedback. Currently, we are particularly interested in the following topics:

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

1.4 Organization

Section 2 outlines a mapping of the potential standardization space, into specification levels of domains, primitives and threshold modes. Section 3 considers application motivations for threshold schemes. Section 4 discusses concrete primitives and threshold modes of interest in the multi-party and in the single-device domains. Section 5 emphasizes several features whose consideration is required when specifying criteria for concrete items. Section 6 discusses the generic phases of development towards 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

To organize the potential space of standardization of threshold schemes, we start by distinguishing two domains: single-device and multi-party. The **single-device** domain is associated with a rigidity of configuration of components, strictly defined physical boundaries, and a dedicated communication network. Conversely, the **multi-party** domain intends to enable modularized patching of components (e.g., repairing newly found bugs in existing components, or even entirely replacing old components by new ones) and may allow dynamic configurations of the parties in a protocol (possibly decided by an administrative authority). The multi-party case may also require solving problems related to distributed systems, such as byzantine agreement (consensus).

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

2.2 Primitives

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

The process of developing new standards must include establishing a clear rationale to support concrete selections. Therefore, it is likely that the first new published standards will stem from simple techniques capable of thresholdizing already 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 n-out-of-n or even k-out-of-n manner. When a cryptographic operation is required, each party individually computes something with their secret share, and later the outputs are combined, without ever combining together the shares that would enable

recovering the secret key. Other simple examples can include threshold schemes resulting from simple combinations of techniques similar or closely related to those standardized, as may happen to achieve some multi-signatures with independent keys.

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

With respect to publishing standards, over time we will reach cases that require more complex compositional design approaches, possibly using some building blocks that do not currently appear in any NIST standard. This is nonetheless focused on schemes with well-understood security properties of the overall design. Since the base primitives of focus are NIST-approved cryptographic primitives, the task of analyzing the security and parameters of the original non-threshold algorithm is likely to not be an hindrance for 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 at the level of the non-threshold primitive. Rather, in such cases the complexity of standardization is in specifying the building blocks, defining a protocol for a chosen threshold mode (see Section 2.3), and analyzing the security of the composition.

2.3 Modes

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

At a high level, we consider a similar paradigm 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. In the perspective of the client, the threshold entity can still be abstracted as a cryptographic module (and in some cases may even be indistinguishable from a conventional one), although possibly with some additional sophistication in the interface and/or on how to interpret the input and output.

We define the *threshold mode* as a level of characterization used to distinguish properties of the threshold scheme in the perspective of the client. Note: the meaning of "mode" here

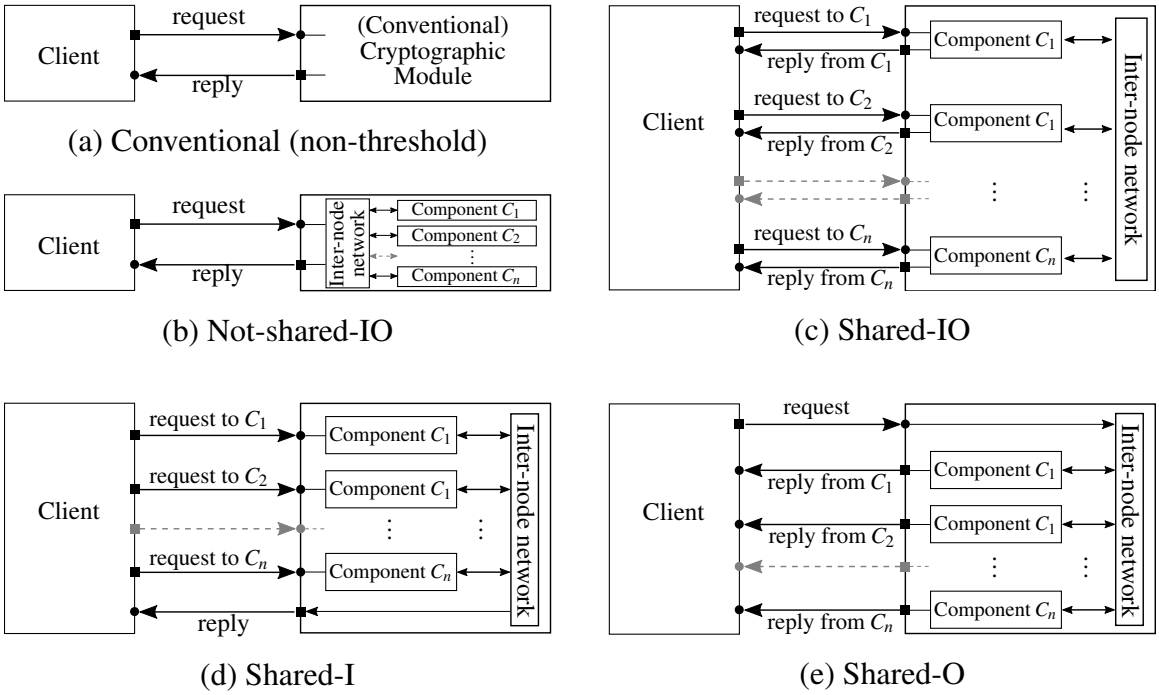

Figure 2. Several threshold interfaces (and one non-threshold case)

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 also depicts several distinct interfaces for the threshold case: no I/O secret-sharing (Figure 2b), secret-sharing of both input and output (Figure 2c), secret-sharing of only the input (Figure 2d), secret-sharing of only the output (Figure 2e). The figures are mere abstractions. The actual communication medium and the input/output connections depend on the implementation and on a more detailed specification of the threshold scheme.

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

- **input/output interface (on the client)** — whether or not the client needs to perform secret sharing of the input and/or secret reconstruction of the output; and
- **auditability** — whether or not the client can prove that an obtained output was produced by a threshold scheme (e.g., identifying k components with registered identities in some public-key infrastructure).

Other threshold mode aspects may be considered along the standardization process.

2.3.1 Input/output interface

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

- **Not-shared-IO:** the client sends to the threshold entity (via a relaying proxy or primary component, or by broadcasting to all components) the full input, and later receives back the output, exactly as in the non-threshold scheme.
- **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.
- **Shared-O:** upon 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.
- **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.

Note: we use "shared-I/O" to denote any case within shared-I, shared-O, and shared-IO.

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

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 client and components). In a not-shared-IO mode the client may or may not be aware of the threshold nature of the cryptographic "module".

Note on other schemes. While some of the shared-I/O modes address privacy concerns about the input or output, there are more sophisticated schemes where not even a full collusion of the components/parties of the threshold scheme would learn anything from the

input. Those schemes, where the client does not let go of the secrecy of the input and output, even if the module is not thresholdized, are possible for example based on secure two-party computation. These schemes fall outside the direct scope of the threshold cryptography project, but are within the area of interest of the privacy-enhancing cryptography project

2.3.2 Auditability

We denote a mode as *auditable* if the client is able to verify and prove to a third party that the obtained result was generated from a threshold execution. This property is for example obvious in a signature defined as a concatenation of signatures, since the client can later show several signed components. Perhaps less obvious, but quite useful, is the case of [concise] multi-signatures whose size is independent of the number of signing parties, and whose verification is similar to that of the non-threshold signature. These schemes define a procedure whereby the client determines an 'equivalent' public-key corresponding to the combination/aggregation of keys of the involved parties, such that a successful signature verification based on the derived public key implies that the several parties have participated.

Auditability may be considered orthogonal to the aspect of I/O interface. For example, a shared-I/O mode does not imply auditability (even though the client uses secret-sharing), since the final reconstructed output may be equal to one from a conventional implementation, without a way to externally prove a threshold computation. A not-shared-IO mode may allow auditability in the case where there is complementary information (e.g., zero-knowledge proofs, or transcripts of authenticated communication with multiple components) allowing verification of the participation of multiple components with registered identities.

2.3.3 Interchangeability

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

3 Motivating applications

The selection of items (primitive–mode) of interest for standardization should consider potential applications taking advantage of threshold schemes for cryptographic primitives. This 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, we consider that a default motivation for thresholdization is the ability to distribute trust across several operators.

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

In the multi-party domain, the distribution of shares across multiple parties can enable removing single points of failure of availability 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, we focus on applications in the active model, where corrupted parties can deviate arbitrarily from the protocol specification. As such, we consider enabling verification of correctness of a produced output (or contributed share). For the single-device domain there is also interest in exploring schemes with active security, but we also see 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 Items across two tracks

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

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

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

We do not assume the following lists to be exhaustive.

4.1 Multi-party track

4.1.1 Simpler cases

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

RSA decryption. We consider the interchangeable mode, which is essentially the same as considered for signatures, except that the input is a ciphertext and the output is a (possibly encoded) plaintext. Since the plaintext is the usual object of confidentiality concerns, for the decryption operation we also envision as potentially relevant the shared-O mode, i.e., as an enhanced way of preventing leakage of sensitive data.

EdDSA signing.[1] The EdDSA is a deterministic variant the Schnorr signature. There are probabilistic Schnorr signatures that can be easily thresholdized, in a simultaneously auditable and interchangeable mode, with the verification key depending on the set of participating signers for each signature, but the signature still being similar in syntax to an original non-threshold signature. The concrete (deterministic) EdDSA replaces the randomness by a hash of the concatenation of the secret signing key and the message being signed. This creates a technical difficulty for achieving a corresponding threshold interchangeable mode, which may either imply for it a more complex longer path of standardization, or additional possible considerations about the exact intended threshold mode.

Key generation for elliptic curve cryptography (ECC). For EdDSA and ECDSA signatures, the secret key is a multiplicative factor (in elliptic curve notation) that leads a public generator into the public key. The generation of secret keys for the mentioned elliptic-curve signatures can be easily performed 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.

4.1.2 More complex cases

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 parties learning secret shares of the secret decryption/signing key d. This can be achieved based on secure multi-party computation, and there are implementations that demonstrate its feasibility.

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

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

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

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

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

4.2.1 Simpler cases

AES enciphering with masked input. 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.

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

Others. It is foreseeable that the insights gained in developing guidelines for implementation and validation of threshold circuit designs for AES may also be applicable to other symmetric-key cryptographic algorithms, e.g., a hash-based message authentication code (HMAC). Public-key cryptography is also implemented in single devices, but as a use-case for threshold circuit design we are comparatively more focused on AES.

4.2.2 More complex cases

Actively secure AES enciphering. Beyond passive security, it is desirable to develop resistance against combined attacks (side-channel and injected faults). This may involve more sophisticated techniques, e.g., producing and distributing correlated randomness, and verifying it, and is therefore considered as 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 (§5.1), configurability and security features (§5.2), and modularity (§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.

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

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

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

5.2 Configurability and security features

Some detailed configuration and security features need to be considered in the phases of defining criteria for calls for contribution, and their evaluation/comparison. Some of them may also depend on more detailed application scenarios to choose as motivation. We describe some important aspects here.

5.2.1 Threshold numbers

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

5.2.2 Rejuvenation of components

In several application settings of threshold schemes, the ability to support rejuvenation of components is essential. Rejuvenations can be proactive or reactive, and parallel or sequential. In the multi-party domain, a rejuvenation may include an actual replacement of a physical machine, or the rebooting of a virtual machine, and may include onboarding the state of the new component. In the single-device setting this may involve redoing a secret sharing of an encryption key.

5.2.3 Advanced security properties

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

- **(Composability)** in which 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?
- **(New properties)** The set of security properties to be required from threshold schemes can be more complex than with the corresponding conventional schemes, and may require some redefinition. For example, in an indistinguishability game for decryption, one may have to count adversarial queries made by isolated components, even if such component is then not part of an actual decryption.

5.3 Modularity

The process of standardizing multiple threshold schemes should consider appropriate trade-offs of construction complexity (from building blocks to complex compositions) and specification detail (from security definitions to concrete instantiations). Figure 3 represents the

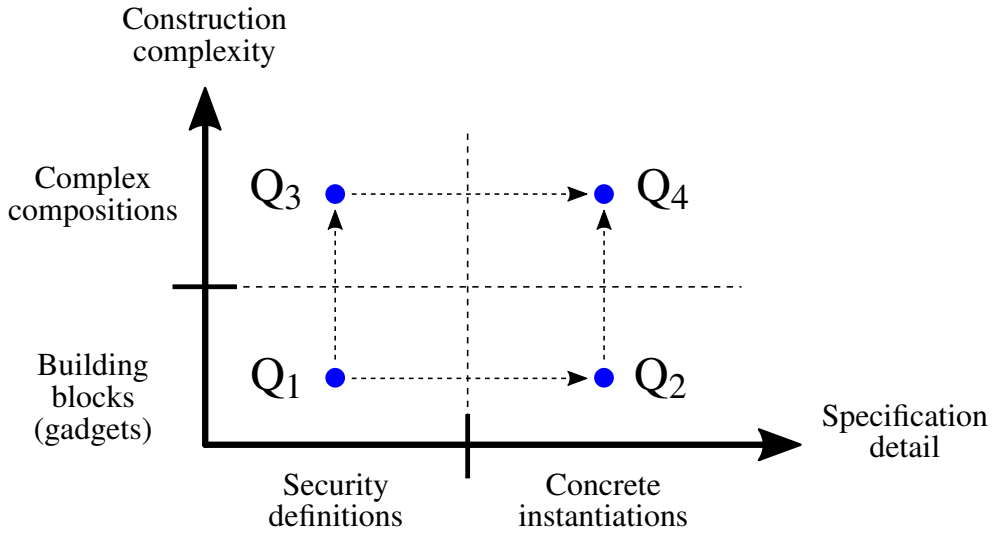

Figure 3. Modularity tradeoffs

abstract states and alternative paths of the evolution process, towards 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 ahead.

5.3.1 Security definitions of building blocks (Q_1)

Reference definitions of abstract gadgets (e.g., such as secret sharing and commitment schemes) can be reused across various threshold schemes, promoting interoperability and alleviating redundant redefinitions. This allows a more modular/compositional description of complex protocols. When incorporating for the first time a gadget into a standard, 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 in consideration.

Secret sharing is a particular case of a gadget applicable across all primitives. Assuming a key has been secret shared, some simple threshold schemes follow in a straightforward manner, using techniques very similar to the original algorithm. Conversely, more complex threshold schemes are likely to benefit from reference definitions of other gadgets, 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 thus consider comparing concrete constructions of gadgets applicable across various threshold schemes. For example, for commitment schemes one can devise guidance on how to implement hash-based commitments and Pedersen commitments, and in which cases each may be preferable, based on comparative advantages.

5.3.3 Security definitions for complex compositions (Q_3)

We want to take advantage of the clarity provided by ideal functionalities, or a defined interface and comprehensive set of security properties. These can be used for defining the threshold modes being sought, and the properties that the corresponding protocols need to satisfy. However, they are 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 of interest for standardization, we want to eventually specify a concrete threshold scheme (Q_4). 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 towards standardization, putting special emphasis of the types of calls for contributions that they may entail. We seek a transparent and open process, involving the community of stakeholders [NISTIR 7977].

We define four generic *phases* towards new standards of threshold schemes:

1. **Roadmap.** Develop a preliminary roadmap (including discussion of this document).
2. **Calls.** Devise calls for contributions, with timelines and criteria for evaluation of input.
3. **Evaluation.** Obtain and evaluate contributions provided upon a call.
4. **Publish.** Write and publish new standards and guidelines

After settling on the preliminary roadmap, the subsequent phases should be tailored independently for each identified standardization item, with separate timelines. For some items, some phases may have several rounds, e.g., possibly alternating several calls for (phase 2) and evaluation of (phase 3) contributions.

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 a preliminary roadmap

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

6.2 Phase 2 — Develop criteria

The NISTIR 8214 has already enumerated several representative questions to consider when reflecting about criteria. To recall, here are some to consider:

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 in consideration, but are not themselves a specification of criteria. In fact, several of them should remain as useful topics of future discussion, besides being recalled here for the purpose of soliciting feedback about them. The goal of phase 2 is to issue criteria, 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.

6.3 Phase 3 — Collect and evaluate 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, we envision different initial types of calls (here described at high level):

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

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

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

We are also interested 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.

We will try to engage with the research community in some appropriate manner (e.g., dedicated workshops), to keep informed about the state-of-the-art in the corresponding fields.

6.4 Phase 4 — Publish new standards

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

7 Collaboration with stakeholders

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

We value the expert technical feedback from stakeholders and will incorporate it in our 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.

Section 6.2 has already mentioned important elements for which we expect useful feedback as collaboration. The following subsections enumerate a few further important aspects, as we move towards issuing criteria for new threshold schemes in each domain.

7.1 Multi-party setting

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

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

7.2 Single-device setting

We are interested 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, non-completeness, ...) possible to achieve in threshold circuit designs.
2. Identify useful construction paradigms for threshold circuit design and identify the gadgets that are useful to implement them.
3. Indicate the models/conditions under which the threshold schemes may enable a higher resistance to side-channel and/or fault attacks, e.g., quantifying the increase in number of traces required for a successful differential power analysis attack.
4. Indicate possible parameters (e.g., masking order, number of shares) for realistic implementations of threshold circuit designs.

References

[ACVP] NIST. *Automated Cryptographic Validation Protocol*. https://github.com/usnistgov/ACVP. Accessed 2019.

[CAVP] NIST. *Cryptographic Algorithm Validation Program*. https://csrc.nist.gov/Projects/cryptographic-algorithm-validation-program. Accessed 2019.

[FIPS 140-2] National Institute of Standards and Technology (NIST). *Security Requirements for Cryptographic Modules. Federal Information Processing Standards Publication 140-2*. May 2001. DOI: 10.6028/NIST.FIPS.140-2.

[FIPS 140-3] National Institute of Standards and Technology (NIST). *Security Requirements for Cryptographic Modules. Federal Information Processing Standards Publication 140-3*. March 2019. DOI: 10.6028/NIST.FIPS.140-3.

[FIPS 186-5] National Institute of Standards and Technology (NIST). *Digital Signature Standard (DSS). Federal Information Processing Standards Publication 186-5*. 2019. DOI: 10.6028/NIST.FIPS.186-5. (To appear).

[FIPS 197] National Institute of Standards and Technology (NIST). *Advanced Encryption Standard (AES). Federal Information Processing Standards Publication 197*. November 2001. DOI: 10.6028/NIST.FIPS.197.

[NISTIR 7977] Cryptographic Technology Group. *NIST Cryptographic Standards and Guidelines Development Process*. NISTIR 7977. March 2016. DOI: 10.6028/NIST.IR.7977.

[NISTIR 8214] Luís T. A. N. Brandão, Nicky Mouha, and Apostol Vassilev. *Threshold Schemes for Cryptographic Primitives: Challenges and Opportunities in Standardization and Validation of Threshold Cryptography*. NISTIR 8214. March 2019. DOI: 10.6028/NIST.IR.8214.

[NTCW 2019] NIST. *NIST Threshold Cryptography Workshop*. https://csrc.nist.gov/Events/2019/NTCW19. Gaithersburg, March 11–12. 2019.

[Proj-LWC] NIST. *Lightweight Cryptography Project*. https://csrc.nist.gov/projects/Lightweight-Cryptography. 2019.

[Proj-PEC] NIST. *Privacy-Enhancing Cryptography Project*. https://csrc.nist.gov/projects/Privacy-Enhancing-Cryptography. 2019.

[Proj-PQC] NIST. *Post-quantum Cryptography Project*. https://csrc.nist.gov/projects/post-quantum-cryptography. 2019.

[Proj-TC] NIST. *Threshold Cryptography Project*. https://csrc.nist.gov/projects/Threshold-Cryptography. 2019.

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

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

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

A Application use cases

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

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

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

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

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

A.2 Protection of secrets at rest

Most cryptographic applications involve a secret, which if revealed to an adversary results in a security failure. For example: a secret key corresponding to a public certificate can decrypt encrypted messages whose content was intended only for the key owner; a secret key from a crypto-currency can be used to spend the original funds of the owner; the secret signing key of a certificate authority (CA) can sign certificates as the CA. The key also needs to 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 above cases, the storage of the secret key in one place represents a *single point of failure* for confidentiality, integrity, or availability. This can be mitigated by using secret sharing to distribute across multiple parties the trust in the storage of secrets. Example use-cases: a CA where the signing key is secret-shared among several employees, such that no single employee alone has access to the key; a "social backup" system for crypto-currency wallets, whereby the user distributes shares of the key to several friends, such that if the user's device is lost or breaks, the user can 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 protection of authenticity of messages, i.e., preventing an attacker from masquerading as Alice to others, with respect to a secret signing key.

Using a threshold decryption (e.g., RSA) in a shared-O mode, the multiple parties compute separate shares of the decryption plaintext, and then a combiner (possibly the end recipient) 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 explicit participation of multiple parties, who can for example log their operations for future audits. Also, in an enhanced auditable mode the recipient of the final decryption can verify which decryptor parties were involved.

A.4 Decentralization of trust for key generation and distribution

Key generation and distribution are essential phases of many cryptographic schemes and applications. For example, a key distribution center (KDC) can act as a trusted service that distributes symmetric secret keys to clients, to enable private communication within groups or to mediate access to other services. A KDC thus represents a single point of failure: if the KDC is offline, clients cannot securely communicate nor access needed services; if it 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 of 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.

A.5 Accountability and prevention of ill-intentioned operations

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

For example, to authorize a large bank transfer, it can be useful to require agreement between several managers. A policy can state that transactions above a certain amount are only valid once 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 enable this in an 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 its own independent secret-public key pair, enabling an auditable mode where it possible to check which managers participated, thereby facilitating accountability.

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 for a threshold RSA signature, each HSM only has to perform an already supported cryptographic operation. Each HSM simply computes and outputs a regular RSA signature, using a signing key share, and then some external non-HSM device combines the output shares to obtain the final RSA signature. This application can be enabled by a dealer that, in an initial safe/protected phase, secret-shares the RSA key, and distributes one share to each HSM (across diverse locations). For more complex threshold schemes (including RSA key generation without a dealer), the threshold operations

may require customized programing and interactions between parties. This can be achieved for example by diverse virtual machines running in various and diversified computers (e.g., with different operating systems and protected by different access control mechanisms).

A.7 Distributed password authentication

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

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

The online attack (issue iii above) can be addressed with extra steps, such as for example: (i) the client sends the password in a shared-I mode, i.e., as separate secret shares to each server; (ii) then the servers, each also with a salt share, jointly compute the salted hash, but without even recombining the salt (efficiency-wise this may benefit from a hash function that is friendly with respect to distributed computations); (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 PRFs. 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.