

Withdrawn Draft

Warning Notice

The attached draft document has been withdrawn, and is provided solely for historical purposes. It has been superseded by the document identified below.

Withdrawal Date July 7, 2020

Original Release Date November 11, 2019

Superseding Document

Status Final

Series/Number NIST Interagency or Internal Report (NISTIR) 8214A

Title NIST Roadmap Toward Criteria for Threshold
Schemes for Cryptographic Primitives

Publication Date July 2020

DOI <https://doi.org/10.6028/NIST.IR.8214A>

CSRC URL <https://csrc.nist.gov/publications/detail/nistir/8214a/final>

Additional Information Threshold Cryptography project

<https://csrc.nist.gov/projects/threshold-cryptography>

1

Draft NISTIR 8214A

2

Towards NIST Standards for Threshold Schemes for Cryptographic Primitives

3

4

A Preliminary Roadmap

5

Luís T. A. N. Brandão

6

Michael Davidson

7

Apostol Vassilev

8

This publication is available free of charge from:

9

<https://doi.org/10.6028/NIST.IR.8214A-draft>

10

11

Draft NISTIR 8214A

12

Towards NIST Standards for Threshold Schemes for Cryptographic Primitives

13

14

A Preliminary Roadmap

15

Luís T. A. N. Brandão

16

Michael Davidson

17

Apostol Vassilev

18

Computer Security Division

19

Information Technology Laboratory

20

This publication is available free of charge from:

21

<https://doi.org/10.6028/NIST.IR.8214A-draft>

22

November 2019

23



24

U.S. Department of Commerce

25

Wilbur L. Ross, Jr., Secretary

26

National Institute of Standards and Technology

27

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

28 National Institute of Standards and Technology Internal Report 8214A
29 36 pages (November 2019)

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

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

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

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

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

46 National Institute of Standards and Technology
47 Attn: Computer Security Division, Information Technology Laboratory
48 100 Bureau Drive (Mail Stop 8930) Gaithersburg, MD 20899-8930
49 Email: nistir-8214A-comments@nist.gov

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

51 **Reports on Computer Systems Technology**

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

60 **Abstract**

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

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

79 **Acknowledgments**

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

86

Call for Patent Claims

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

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

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

97 b) assurance that a license to such essential patent claim(s) will be made available to applicants desiring
98 to utilize the license for the purpose of complying with the guidance or requirements in this ITL draft
99 publication either:

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

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

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

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

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

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

113 **Executive Summary**

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

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

134 This document sets a preliminary roadmap towards the standardization by NIST of
135 threshold schemes for cryptographic primitives. This phase follows the publication of
136 the NIST Internal Report “Threshold Schemes for Cryptographic Primitives” ([NISTIR
137 8214](#)), which positioned a preparatory framework and several representative questions, and
138 the “NIST Threshold Cryptography Workshop” ([NTCW](#)) 2019, which brought together
139 stakeholders to share perspectives from industry, academia and government.

140 The positive feedback received on the report (NISTIR 8214) and on the workshop
141 (NTCW 2019) confirms that there is interest and adequate knowledge by the stakeholders to
142 initiate the process of standardization of threshold schemes. To prepare such an endeavor,
143 this document tackles the challenge of differentiating various aspects of the standardization
144 effort, while simultaneously aiming to enable an open and transparent process with the
145 collaboration of the community of stakeholders. This document thus defines the approaches
146 to devise criteria for future multiple open calls for contributions for standardization, with
147 a focus on NIST-approved primitives. This provides a number of opportunities but also
148 requires dealing with a number of challenges.

149 The main challenge is devising an effective mechanism to navigate through the large
150 diversity of possible threshold schemes, namely to organize, prioritize, and engage with the
151 stakeholders for collaboration and feedback. To this effect, this document starts by orga-

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

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

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

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

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

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

189	Table of Contents		
190	1	Introduction	1
191	1.1	A multifaceted standardization effort	1
192	1.2	A structured approach	2
193	1.3	Feedback from stakeholders	4
194	1.4	Organization	4
195	2	The space of threshold schemes for potential standardization	5
196	2.1	Two domains	5
197	2.2	Primitives	5
198	2.3	Modes	6
199	3	Motivating applications	10
200	4	Items across two tracks	11
201	4.1	Multi-party track	11
202	4.2	Single-device track	13
203	5	Features of standardization items	15
204	5.1	Validation suitability	15
205	5.2	Configurability and security features	15
206	5.3	Modularity	16
207	6	Development phases	19
208	6.1	Phase 1 — Develop a preliminary roadmap	19
209	6.2	Phase 2 — Develop criteria	19
210	6.3	Phase 3 — Collect and evaluate contributions	20
211	6.4	Phase 4 — Publish new standards	21
212	7	Collaboration with stakeholders	22
213	7.1	Multi-party setting	22
214	7.2	Single-device setting	22
215	A	Application use cases	25
216	A.1	Single-device encryption resistant to side-channel attacks	25
217	A.2	Protection of secrets at rest	25
218	A.3	Confidential communication	26
219	A.4	Decentralization of trust for key generation and distribution	26
220	A.5	Accountability and prevention of ill-intentioned operations	27
221	A.6	Distribution of trust across secure environments	27
222	A.7	Distributed password authentication	28
223	List of Figures		
224	1	A depiction of a variety of primitives and threshold modes across two domains	3
225	2	Several threshold interfaces (and one non-threshold case)	7
226	3	Modularity tradeoffs	17

227 1 Introduction

228 NIST has established the [Threshold Cryptography project](#) to drive an effort to standardize
229 threshold schemes for cryptographic primitives. Threshold schemes enable distribution of
230 trust placed on human operators, and also offer a path to prevent several single-points of
231 failure in conventional cryptographic implementations. This document comes on the heels
232 of the [NIST Internal Report \(NISTIR\) 8214](#), which posed representative questions about
233 standardization of threshold schemes, and the [NIST Threshold Cryptography Workshop](#)
234 [\(NTCW\) 2019](#), which brought together a variety perspectives from stakeholders.

235 The NISTIR 8214 had already identified the need to devise criteria for eventual calls
236 for contributions for the development of new standards of threshold cryptographic schemes.
237 This document (NISTIR 8214A) is intended to devise a preliminary roadmap for the stan-
238 dardization effort. A main motivation is to lay out reference rationale (complementary to
239 what the NISTIR 8214 has already done), terminology, and structure that are conducive, as
240 the project moves forward, to a precise description of the material to standardize. This is still
241 an early step that identifies at a high level the space of standardization, and a corresponding
242 variety of manners to approach possible items, with possible different timelines.

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

251 1.1 A multifaceted standardization effort

252 **Diverse stakeholders.** The challenge inherent to this standardization endeavor goes be-
253 yond the technical considerations about the simple and the sophisticated algorithms and
254 techniques that enable threshold schemes for some cryptographic primitives. We recognize
255 a diverse set of stakeholders, including not only experts in the field of threshold cryptog-
256 raphy, but also users, vendors, security practitioners, and those who work in cryptographic
257 standards but may be unfamiliar with threshold techniques. The structure in this document
258 is intended to engage all stakeholders and generate feedback about the roadmap ahead.

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

266 **On a variety of goals and paths.** As the field of threshold schemes encompasses many
267 possibilities, we consider several approaches, not all of which fall within the scope of
268 developing new standards. For standardization, we are focused on threshold schemes for
269 NIST-approved cryptographic primitives. We want to enable the standardization of threshold
270 modes of implementation for these primitives, as a way to promote better best practices in
271 settings where the use of these primitives is considered to be subject to adversarial attacks
272 on the implementation or on the operation.

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

278 This effort will inevitably lead to some open problems of interest to the research com-
279 munity. For example, threshold versions of candidate primitives under current evaluation
280 within other NIST projects, such as the [post-quantum cryptography](#) and the [lightweight](#)
281 [cryptography](#), where the proposed conventional non-threshold primitives are still under
282 security evaluation. Although interesting, these cases are not considered here as in scope
283 for standardization. Nonetheless, there is interest in learning about new research results and
284 developments in the state of the art.

285 **On the types of standard/documents to produce.** For some of the items identified in this
286 document, a natural question is: *do we need a standard for this?* The question leaves implicit
287 the meaning of *standard*, which may vary with the context. In some cases a reasonable end
288 goal may be to add a simple addendum (e.g., of a simple threshold mode) in an existing
289 standard; in others an appropriate goal may be to devise reference definitions (e.g., of secret
290 sharing) that may appear as building block of several new techniques to consider; in some
291 other cases a worthy goal may be to devise implementation guidelines that enable validation
292 within a certain security profile level that confirms certain threshold properties; in some
293 cases we may actually consider specifying particular new algorithms. The concrete form
294 in which to deliver the new standards will become apparent as we move forward.

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

297 **1.2 A structured approach**

298 **1.2.1 The potential space of standardization**

299 Since the space of threshold schemes has many dimensions, the analysis of potential items
300 for standardization benefits from a structured approach. We start by distinguishing the
301 single-device and multi-party *domains*. In each domain there is a potential applicability for
302 several cryptographic *primitives*, and each of those can be potentially implemented in various
303 *modes*. However, not every conceivable possibility is suitable for standardization. Simplicity
304 of standardization does not necessarily imply that an item should be standardized. Similarly,

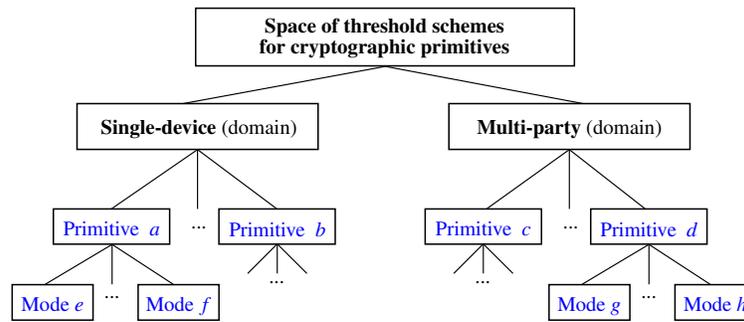


Figure 1. A depiction of a variety of primitives and threshold modes across two domains

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

307 1.2.2 Motivating applications

308 While there are many conceivable threshold schemes, we consider important to focus on
309 where there is a high need and high potential for adoption. An overarching motivation in
310 this effort is developing the ability to distribute trust in operations, and increasing resistance
311 against attacks on implementations, of NIST-approved cryptographic primitives, since they
312 already underpin the security of many real systems. Several potential applications can benefit
313 directly from the threshold properties enabled in implementations of these cryptographic
314 primitives. We can benefit in learning from stakeholders about more concrete applications.

315 1.2.3 Items across two tracks

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

320 For each track we are interested in organizing possible items (primitive/mode) for
321 standardization. Some of the default potential primitives to consider for thresholdization
322 come from NIST standards specifying the Rivest–Shamir–Adleman (RSA) signature and
323 encryption schemes, the Elliptic Curve Digital Signature Algorithm (ECDSA), the Edwards
324 Curve Digital Signature Algorithm (EdDSA), the Advanced Encryption Standard (AES), and
325 methods for random number generation (RNG). Within these, there is a special interest in the
326 primitives related to secret keys, such as key-generation, signing, decryption within a public-
327 key encryption (PKE) scheme, and symmetric-key enciphering and deciphering. For each
328 primitive we are interested in considering what are the relevant threshold modes of operation,
329 and how some of their technical challenges may vary with respect to standardization.

330 1.2.4 Detailed features

331 Besides the high level identification of threshold modes of interest, there are detailed features
332 of fundamental importance in the upcoming phase of criteria definition. This preliminary

333 roadmap emphasizes three aspects: *configurability and security features* — need to be
334 specified in order to characterize the threshold scheme, including its interface; *suitability*
335 *for validation* — required in the process of allowing the use of cryptographic schemes in
336 several application scenarios (e.g., in the U.S. federal context); *modularity of components*
337 *and specification detail* — relevant to identify recurring building blocks (such as secret
338 sharing) that may appear across several threshold schemes, as well as improving the security
339 analysis and the simplicity of specification.

340 **1.2.5 Development phases**

341 We intend to drive the standardization project in phases of devising criteria for calls for con-
342 tributions, evaluating proposed contributions, and writing documentation for new standards.
343 Standardization items with different development needs may be organized into different
344 tailored calls for contributions and corresponding timelines. This improves collaboration
345 with a set of stakeholders interested in a variety of standardization items and challenges.
346 Expected new standards and guidelines may include reference definitions (e.g., for secret
347 sharing), algorithms/techniques for threshold implementations, and security profiles for
348 validation/certification. The resulting documentation may span a variety of formats, includ-
349 ing addenda to existing standards (e.g., a simple threshold mode of operation), and new
350 standalone documents (e.g., describing new complex techniques and analysis).

351 **1.3 Feedback from stakeholders**

352 To drive an open and transparent standardization process, the several phases present oppor-
353 tunities for public feedback. Currently, we are particularly interested in the following topics:

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

359 **1.4 Organization**

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

368 2 The space of threshold schemes for potential standardization

369 2.1 Two domains

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

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

384 2.2 Primitives

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

398 The process of developing new standards must include establishing a clear rationale to
399 support concrete selections. Therefore, it is likely that the first new published standards will
400 stem from simple techniques capable of thresholdizing already NIST-approved algorithms.
401 One probable example, simple and concrete, is that of a threshold version of RSA signing or
402 decryption, where the private RSA key is initially secret-shared across several parties. This
403 can be instantiated in a n -out-of- n or even k -out-of- n manner. When a cryptographic oper-
404 ation is required, each party individually computes something with their secret share, and
405 later the outputs are combined, without ever combining together the shares that would enable

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

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

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

426 2.3 Modes

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

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

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

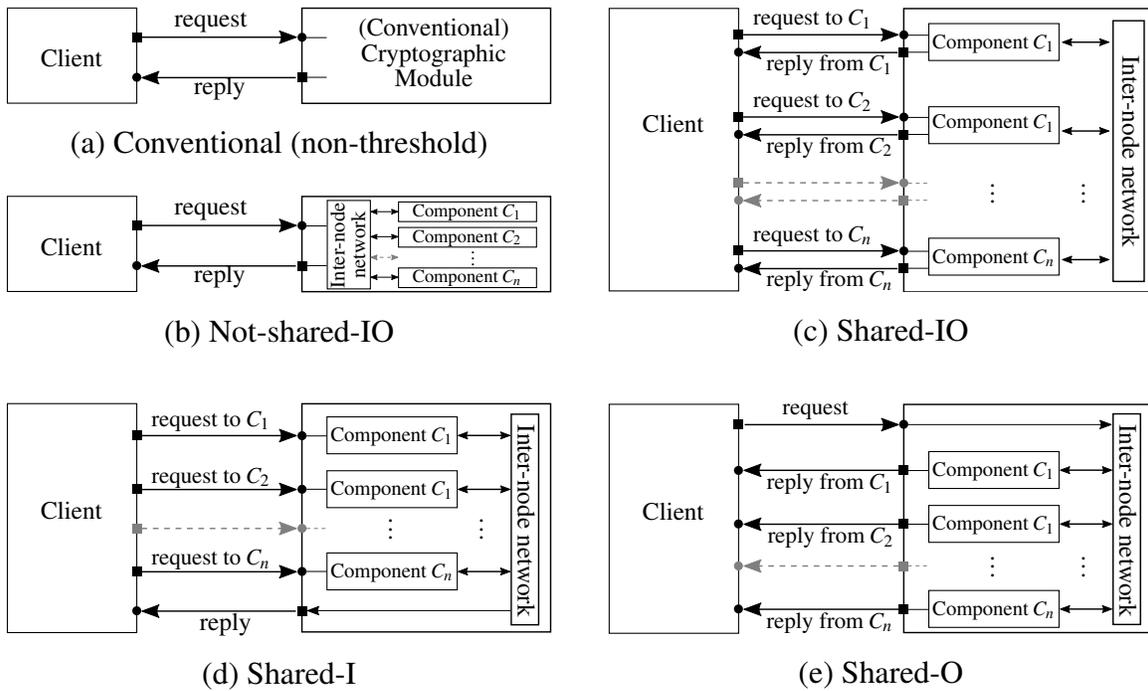


Figure 2. Several threshold interfaces (and one non-threshold case)

444 should not be confused with the usage in “block-cipher mode of operation”, which identifies
 445 how a block-cipher can be used to encrypt and decrypt large messages.

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

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

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

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

458 2.3.1 Input/output interface

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

- 460 • **Not-shared-IO:** the client sends to the threshold entity (via a relaying proxy or
461 primary component, or by broadcasting to all components) the full input, and later
462 receives back the output, exactly as in the non-threshold scheme.
- 463 • **Shared-I:** the client secret-shares the input in a k -out-of- n manner; and then sends
464 each share to each component of the threshold scheme; the components may then
465 communicate between themselves to securely compute the output (e.g., a ciphertext c)
466 without learning the input. This mode is relevant for enhanced secrecy of the input,
467 e.g., a plaintext submitted for symmetric encryption, or possibly even for signing.
- 468 • **Shared-O:** upon a threshold computation, each component obtains only a secret share
469 of the output (e.g., of a decrypted plaintext), and sends it to the client; the client then
470 reconstructs the final output from the shares. This mode is relevant for enhanced
471 secrecy of output, e.g., a plaintext obtained from threshold decryption.
- 472 • **Shared-IO:** both the input (I) and the output (O) are secret-shared across the com-
473 ponents of the threshold scheme. Only the client sees the complete input and output.
- 474 Note: we use “shared-I/O” to denote any case within shared-I, shared-O, and shared-IO.

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

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

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

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

501 **2.3.2 Auditability**

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

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

518 **2.3.3 Interchangeability**

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

3 Motivating applications

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

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

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

547 In the multi-party domain, the distribution of shares across multiple parties can enable re-
548 moving single points of failure of availability by not requiring all parties to be present, of con-
549 fidentiality by requiring a greater number of colluding parties to find the key, and of integrity
550 by implementing robust techniques that detect and address faults from malicious parties.

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

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

561 Appendix A describes potential application use-cases, such as: single-device encryption
562 resistant to side-channel attacks; protection of secrets at rest; trust decentralization for key
563 generation and distribution; accountability and prevention of ill-intentioned operations; confi-
564 dential communication; password authentication; and interacting hardware security modules.

565 4 Items across two tracks

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

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

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

586 We do not assume the following lists to be exhaustive.

587 4.1 Multi-party track

588 4.1.1 Simpler cases

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

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

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

609 **Key generation for elliptic curve cryptography (ECC).** For EdDSA and ECDSA sig-
610 natures, the secret key is a multiplicative factor (in elliptic curve notation) that leads a public
611 generator into the public key. The generation of secret keys for the mentioned elliptic-curve
612 signatures can be easily performed from independent random shares. To ensure that each
613 party ends with an actual random share, the distributed key generation may also include
614 multiparty coin-flipping and commitments to the shares held by every party.

615 **4.1.2 More complex cases**

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

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

627 **AES enciphering and deciphering.** The mathematical structure of the AES S-Box (the
628 non-linear component of AES) does not provide homomorphic properties enabling an
629 easy thresholdization in the multi-party setting. Nonetheless, threshold versions can be
630 implemented based on techniques of secure multiparty computation. Threshold versions
631 of enciphering and deciphering can be of interest in the [shared-I](#) and [shared-O](#) modes,
632 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.

633 4.2 Single-device track

634 Historically, cryptographic algorithms were implemented in hardware devices long before
635 cryptography appeared in software. As software cryptographic implementations started to
636 dominate the mainstream technology used at home and the office, people again turned to
637 hardware for acceleration and security. For example, AES instructions and Secure Hash
638 Algorithm (SHA) extensions were provided on Intel x86, AMD and ARM processors. More
639 recently, as the complexity of single-chip devices increased and the emergence of Systems
640 on a Chip (SoC) technology became mainstream, more complete implementations of crypto-
641 graphic capabilities appeared in hardware. For example, the rapid and accelerating growth of
642 Field Programmable Gate Arrays (FPGA) devices in recent years in response to existing and
643 emerging computational needs in different domains, including deep learning and artificial
644 intelligence, bring opportunities in using the FPGA platform as both an accelerator for
645 cryptographic algorithms and as a host platform with cryptographic capabilities intended
646 to protect the intellectual property of the customization logic programmed on the platform.

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

654 4.2.1 Simpler cases

655 **AES enciphering with masked input.** Leakage resilience can be achieved based on
656 masking techniques for generic Boolean circuits. This involves a secret-sharing of the input
657 key material so that each wire or register only “sees” a share, and never an actual secret bit.
658 Furthermore, the protection needs to be propagated across the circuit path, in order to prevent
659 leakage of sensitive internal states of the computation. Under certain attack models, the
660 number of side-channel traces that need to be collected is exponential in the number of shares.

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

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

670 **4.2.2 More complex cases**

671 **Actively secure AES enciphering.** Beyond passive security, it is desirable to develop re-
672 sistance against combined attacks (side-channel and injected faults). This may involve more
673 sophisticated techniques, e.g., producing and distributing correlated randomness, and verify-
674 ing it, and is therefore considered as more complex. Ways of achieving this include crypto-
675 graphic checksums (such as message authentication codes), whose result cannot be predicted
676 by an adversary with only a partial view of the internal state. To be pertinent these schemes
677 should be demonstrably better than a simple redundant execution of the circuit computation.

678 **5 Features of standardization items**

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

683 **5.1 Validation suitability**

684 The process of standardizing new threshold schemes entails devising corresponding testing
685 and validation requirements, which may differ from those for conventional implementations.
686 This applies both to validation of modules and validation of the algorithms therein.

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

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

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

708 **5.2 Configurability and security features**

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

713 5.2.1 Threshold numbers

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

719 5.2.2 Rejuvenation of components

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

726 5.2.3 Advanced security properties

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

- 732 • **(Composability)** in which way does security remain when the scheme is composed
733 with other protocols, including in concurrent executions, possibly depending on the
734 actual instantiation of a required trusted setup?
- 735 • **(Adaptive security)** is the adversary allowed to observe the protocol execution before
736 deciding which components to corrupt?
- 737 • **(Graceful degradation)** is there a controlled vs. uncontrolled breakdown as soon as
738 the threshold number of corruptions is surpassed?
- 739 • **(New properties)** The set of security properties to be required from threshold schemes
740 can be more complex than with the corresponding conventional schemes, and may
741 require some redefinition. For example, in an indistinguishability game for decryption,
742 one may have to count adversarial queries made by isolated components, even if such
743 component is then not part of an actual decryption.

744 5.3 Modularity

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

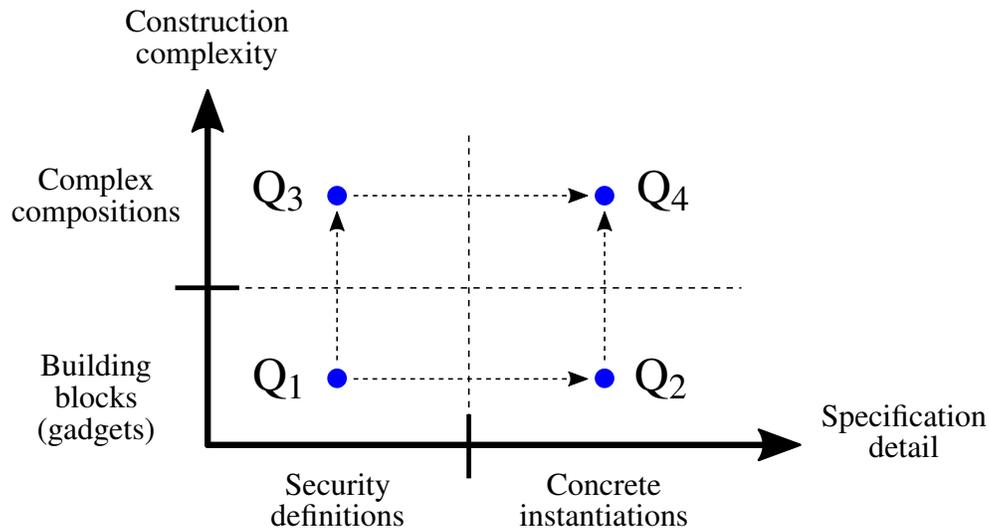


Figure 3. Modularity tradeoffs

748 abstract states and alternative paths of the evolution process, towards obtaining standardized
 749 threshold schemes that are concrete and provably secure instantiations of compositions of
 750 well understood building blocks. The figure shows four symbolic quadrants, explained ahead.

751 5.3.1 Security definitions of building blocks (Q_1)

752 Reference definitions of abstract gadgets (e.g., such as secret sharing and commitment
 753 schemes) can be reused across various threshold schemes, promoting interoperability and
 754 alleviating redundant redefinitions. This allows a more modular/compositional description
 755 of complex protocols. When incorporating for the first time a gadget into a standard, the
 756 gadget should have a well defined interface specified in that standard. This makes it possible
 757 that future standards refer to such descriptions based only on the corresponding interface
 758 and security properties. Some other examples of gadgets may include *consensus*, *generation*
 759 *of correlated randomness*, *reliable broadcast*, *oblivious transfer*, and *garbled circuits*. Their
 760 treatment as modules alleviates the burden of compiling from scratch arguments about the
 761 security of a more complex concrete protocol based on them, provided that composability
 762 properties are taken in consideration.

763 Secret sharing is a particular case of a gadget applicable across all primitives. Assuming
 764 a key has been secret shared, some simple threshold schemes follow in a straightforward
 765 manner, using techniques very similar to the original algorithm. Conversely, more complex
 766 threshold schemes are likely to benefit from reference definitions of other gadgets, since they
 767 may be substantially different from the baseline cryptographic primitive being thresholdized.

768 5.3.2 Concrete instantiations of building blocks (Q_2)

769 The optimized low-level specification of a gadget, such as a commitment scheme, can
770 vary across concrete protocols. Useful guidance may thus consider comparing concrete
771 constructions of gadgets applicable across various threshold schemes. For example, for
772 commitment schemes one can devise guidance on how to implement hash-based commit-
773 ments and Pedersen commitments, and in which cases each may be preferable, based on
774 comparative advantages.

775 5.3.3 Security definitions for complex compositions (Q_3)

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

781 5.3.4 Concrete instantiations of complex compositions (Q_4)

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

6 Development phases

788 This section discusses the possible development phases towards standardization, putting
789 special emphasis of the types of calls for contributions that they may entail. We seek a
790 transparent and open process, involving the community of stakeholders [NISTIR 7977].

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

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

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

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

- 801 a. produce draft documentation and call for feedback;
- 802 b. evaluate and integrate external feedback;
- 803 c. publish documentation.

6.1 Phase 1 — Develop a preliminary roadmap

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

6.2 Phase 2 — Develop criteria

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

- 814 1. definition of system model and threat model;
- 815 2. description of characterizing features;
- 816 3. analysis of efficiency and practical feasibility;
- 817 4. existence of open-source reference implementations;
- 818 5. concrete benchmarking (threshold vs. conventional; different platforms);
- 819 6. detailed description of operations;
- 820 7. example application scenarios;

- 821 8. security analysis (see also Section 5.2);
822 9. automated testing and validation of implementations (see also Section 5.1);
823 10. disclosure and licensing of intellectual property.

824 The above items are important factors to take in consideration, but are not themselves
825 a specification of criteria. In fact, several of them should remain as useful topics of future
826 discussion, besides being recalled here for the purpose of soliciting feedback about them.
827 The goal of phase 2 is to issue criteria, refined per standardization *item*. However, such
828 criteria will only emerge after consideration of feedback from stakeholders, and may happen
829 with different timelines for different items. Furthermore, certain aspects have a life span
830 that goes beyond the initial (future) issuance of criteria. This is for example the case of
831 performing benchmarks, collecting reference implementations developed by the community,
832 and developing testing and validation procedures. The development of these continues after
833 the selection of concrete threshold schemes in subsequent phases.

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

835 6.3 Phase 3 — Collect and evaluate contributions

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

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

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

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

852 We are also interested in research results about useful threshold schemes that are out
853 of scope for this standardization effort. For the multi-party setting, this includes schemes for
854 post-quantum public-key encryption (i.e., their decryption and key-generation algorithms)
855 and signatures. For the single-device setting, this may conceivably include schemes for
856 threshold enciphering, authenticated encryption with associated data (AEAD) and/or hashing
857 related to lightweight cryptographic schemes being currently evaluated.

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

860 **6.4 Phase 4 — Publish new standards**

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

864 In some cases, a simple addendum to an existing standard may be sufficient to define
865 the new mode or modes of threshold operation. For example, for some threshold circuit
866 designs, the standardization of the technique may correspond to defining guidelines with
867 implementation requirements to achieve certification at some security level. For other items,
868 the standardization may result into a new standalone standard.

869 **7 Collaboration with stakeholders**

870 As an immediate followup to this roadmap, we want to solicit specific feedback on the cri-
871 teria for subsequent calls for contributions. To this effect, it is important to obtain feedback
872 from stakeholders about the security definitions and interfaces (and/or ideal functionalities)
873 (see Q₃) upon which protocols/techniques should be evaluated.

874 We value the expert technical feedback from stakeholders and will incorporate it in our
875 standardization process. Along the way, future NIST Threshold Cryptography Workshops
876 (NTCW) may constitute an essential way to obtain interactive public feedback. This can
877 be a place to discuss evaluations about contributions made thus far within the standard-
878 ization process, while covering a variety of approaches across the different domains, and
879 considering distinguished features of interest across various items.

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

883 **7.1 Multi-party setting**

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

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

894 **7.2 Single-device setting**

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

- 897 1. Enumerate and define the desirable properties (e.g., uniformity, non-completeness, ...)
898 possible to achieve in threshold circuit designs.
- 899 2. Identify useful construction paradigms for threshold circuit design and identify the
900 gadgets that are useful to implement them.
- 901 3. Indicate the models/conditions under which the threshold schemes may enable a
902 higher resistance to side-channel and/or fault attacks, e.g., quantifying the increase in
903 number of traces required for a successful differential power analysis attack.
- 904 4. Indicate possible parameters (e.g., masking order, number of shares) for realistic
905 implementations of threshold circuit designs.

906 **References**

- 907 [ACVP] NIST. *Automated Cryptographic Validation Protocol*. [https://github.com/](https://github.com/usnistgov/ACVP)
908 [usnistgov/ACVP](https://github.com/usnistgov/ACVP). Accessed 2019.
- 909 [CAVP] NIST. *Cryptographic Algorithm Validation Program*. [https://csrc.nist.gov/](https://csrc.nist.gov/Projects/cryptographic-algorithm-validation-program)
910 [Projects/cryptographic-algorithm-validation-program](https://csrc.nist.gov/Projects/cryptographic-algorithm-validation-program). Accessed 2019.
- 911 [FIPS 140-2] National Institute of Standards and Technology (NIST). *Security Require-*
912 *ments for Cryptographic Modules. Federal Information Processing Stan-*
913 *dards Publication 140-2*. May 2001. DOI: [10.6028/NIST.FIPS.140-2](https://doi.org/10.6028/NIST.FIPS.140-2).
- 914 [FIPS 140-3] National Institute of Standards and Technology (NIST). *Security Require-*
915 *ments for Cryptographic Modules. Federal Information Processing Stan-*
916 *dards Publication 140-3*. March 2019. DOI: [10.6028/NIST.FIPS.140-3](https://doi.org/10.6028/NIST.FIPS.140-3).
- 917 [FIPS 186-5] National Institute of Standards and Technology (NIST). *Digital Signature*
918 *Standard (DSS). Federal Information Processing Standards Publication*
919 *186-5*. 2019. DOI: [10.6028/NIST.FIPS.186-5](https://doi.org/10.6028/NIST.FIPS.186-5). (To appear).
- 920 [FIPS 197] National Institute of Standards and Technology (NIST). *Advanced En-*
921 *ryption Standard (AES). Federal Information Processing Standards Pub-*
922 *lication 197*. November 2001. DOI: [10.6028/NIST.FIPS.197](https://doi.org/10.6028/NIST.FIPS.197).
- 923 [NISTIR 7977] Cryptographic Technology Group. *NIST Cryptographic Standards and*
924 *Guidelines Development Process*. NISTIR 7977. March 2016. DOI: [10.](https://doi.org/10.6028/NIST.IR.7977)
925 [6028/NIST.IR.7977](https://doi.org/10.6028/NIST.IR.7977).
- 926 [NISTIR 8214] Luís T. A. N. Brandão, Nicky Mouha, and Apostol Vassilev. *Threshold*
927 *Schemes for Cryptographic Primitives: Challenges and Opportunities in*
928 *Standardization and Validation of Threshold Cryptography*. NISTIR 8214.
929 March 2019. DOI: [10.6028/NIST.IR.8214](https://doi.org/10.6028/NIST.IR.8214).
- 930 [NTCW 2019] NIST. *NIST Threshold Cryptography Workshop*. [https://csrc.nist.gov/](https://csrc.nist.gov/Events/2019/NTCW19)
931 [Events/2019/NTCW19](https://csrc.nist.gov/Events/2019/NTCW19). Gaithersburg, March 11–12. 2019.
- 932 [Proj-LWC] NIST. *Lightweight Cryptography Project*. [https://csrc.nist.gov/projects/](https://csrc.nist.gov/projects/Lightweight-Cryptography)
933 [Lightweight-Cryptography](https://csrc.nist.gov/projects/Lightweight-Cryptography). 2019.
- 934 [Proj-PEC] NIST. *Privacy-Enhancing Cryptography Project*. [https://csrc.nist.gov/](https://csrc.nist.gov/projects/Privacy-Enhancing-Cryptography)
935 [projects/Privacy-Enhancing-Cryptography](https://csrc.nist.gov/projects/Privacy-Enhancing-Cryptography). 2019.
- 936 [Proj-PQC] NIST. *Post-quantum Cryptography Project*. [https://csrc.nist.gov/projects/](https://csrc.nist.gov/projects/post-quantum-cryptography)
937 [post-quantum-cryptography](https://csrc.nist.gov/projects/post-quantum-cryptography). 2019.
- 938 [Proj-TC] NIST. *Threshold Cryptography Project*. [https://csrc.nist.gov/projects/](https://csrc.nist.gov/projects/Threshold-Cryptography)
939 [Threshold-Cryptography](https://csrc.nist.gov/projects/Threshold-Cryptography). 2019.

- 940 [SP 800-56A] Elaine Barker, Lily Chen, Allen Roginsky, Apostol Vassilev, and Richard
941 Davis. *Recommendation for Pair-Wise Key-Establishment Schemes Using*
942 *Discrete Logarithm Cryptography. Special Publication 800-56A Rev. 3.*
943 April 2018. DOI: [10.6028/NIST.SP.800-56Ar3](https://doi.org/10.6028/NIST.SP.800-56Ar3).
- 944 [SP 800-56B] Elaine Barker, Lily Chen, Allen Roginsky, Apostol Vassilev, Richard
945 Davis, Scott Simon. National Institute of Standards, and Technology
946 (NIST). *Recommendation for Pair-Wise Key-Establishment Using Integer*
947 *Factorization Cryptography. Special Publication 800-56B Rev. 2.* March
948 2019. DOI: [10.6028/NIST.SP.800-56Br2](https://doi.org/10.6028/NIST.SP.800-56Br2).
- 949 [SP 800-90A] Elaine Barker and John Kelsey. *Recommendation for Random Number*
950 *Generation Using Deterministic Random Bit Generators. Special Publi-*
951 *cation 800-90A Rev. 1.* National Institute of Standards and Technology
952 (NIST). June 2015. DOI: [10.6028/NIST.SP.800-90Ar1](https://doi.org/10.6028/NIST.SP.800-90Ar1).

953 A Application use cases

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

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

958 The hardware implementation of cryptographic algorithms has gained a significant and grow-
959 ing stake in the industry. Large amounts of sensitive data are now processed in hardware,
960 which creates the need for faster implementations. Most semiconductor manufacturers have
961 incorporated dedicated hardware accelerators for cryptography that perform orders of mag-
962 nitude faster than software implementations. Even though asymmetric algorithms, such as
963 RSA and even ECC digital signatures, can be implemented by a hardware accelerator, in or-
964 der to reduce the processing time of private key operations, these algorithms are not suitable
965 for severely constrained devices in the Internet of Things (IoT), due to the significant re-
966 sources required, which results in low performance on such platforms. As a result, many IoT
967 devices have only hardware engines for symmetric cryptography primitives, such as AES.

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

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

982 A.2 Protection of secrets at rest

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

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

999 **A.3 Confidential communication**

1000 For secure communications it is essential to ensure that secret messages are only decrypted
1001 by legitimate recipients. An attacker who steals Alice’s secret decryption key can read
1002 messages intended for Alice. Threshold decryption can help protect confidentiality. It
1003 can for example be used across devices, analogously to multi-factor “authentication” for
1004 a single person, such that unauthorized parties (in this case hacked or stolen devices) cannot
1005 break the confidentiality of messages, without using multiple shares of the key. Similar
1006 considerations apply to protection of authenticity of messages, i.e., preventing an attacker
1007 from masquerading as Alice to others, with respect to a secret signing key.

1008 Using a threshold decryption (e.g., RSA) in a **shared-O** mode, the multiple parties
1009 compute separate shares of the decryption plaintext, and then a combiner (possibly the end
1010 recipient) receives the shares and computes the plaintext from them. This mode of operation
1011 protects the secrecy of the (distributed) key (as a main feature) as well as the confidentiality
1012 of the decrypted message (as an added feature). In some settings this may provide a kind
1013 of accountability, since it requires explicit participation of multiple parties, who can for
1014 example log their operations for future audits. Also, in an enhanced **auditable** mode the
1015 recipient of the final decryption can verify which decryptor parties were involved.

1016 **A.4 Decentralization of trust for key generation and distribution**

1017 Key generation and distribution are essential phases of many cryptographic schemes and
1018 applications. For example, a key distribution center (KDC) can act as a trusted service that
1019 distributes symmetric secret keys to clients, to enable private communication within groups
1020 or to mediate access to other services. A KDC thus represents a single point of failure: if
1021 the KDC is offline, clients cannot securely communicate nor access needed services; if it
1022 is hacked, the attacker can learn the secret keys in use by clients, and can obtain tickets
1023 to access any services. The same considerations apply for example to an identity-based
1024 encryption scheme, where a trusted server holds the master key that is required to generate
1025 a new secret key for every new member (identity) in an organization. Yet another example
1026 is the use of a “dealer” as a trusted party generating a secret key (possibly with a complex
1027 structure, such as an RSA key), only to then secret share it across multiple parties of a
1028 subsequent threshold signing of decryption scheme.

1029 To eliminate this single point of failure, a set of servers can jointly act as a KDC or dealer
1030 in a way that no individual server knows any of the secret keys, and so that services remain
1031 available as long as a certain threshold number of servers have not been hacked or taken
1032 offline. This threshold property can be based on distributed key generation and use of secret
1033 sharing, possibly with proactive and verifiable properties. The latter properties allow the
1034 servers to jointly refresh the secret shares (in order to recover from the potential compromise
1035 of some servers) and to ensure that their shares are consistent. The distribution of servers
1036 prevents any server from learning any master secret key, while the actual distribution of new
1037 keys may fit within a [shared-O](#) mode, so that no server learns any new secret key.

1038 **A.5 Accountability and prevention of ill-intentioned operations**

1039 Entrusting a single individual with the ability to decrypt or sign a message may invite foul
1040 play, if the result cannot be externally verified as correct or its computation does not require
1041 agreement between multiple parties.

1042 For example, to authorize a large bank transfer, it can be useful to require agreement be-
1043 tween several managers. A policy can state that transactions above a certain amount are only
1044 valid once signed off by at least two out of three bank managers, to prevent the authorization
1045 of errant transfers intended by a single ill-intentioned manager. Certain threshold signature
1046 schemes enable this in an [interchangeable](#) mode, such that the output is syntactically equiva-
1047 lent to an original signature — this property can be important for records where size matters
1048 (e.g., storage in a blockchain) and where the policy on the number of signers may be dynamic.
1049 If a single original signing key was secret-shared between the managers, then the bank can in-
1050 ternally know that a large enough subset of managers got together, though possibly not know-
1051 ing (from the signature itself) which ones. If a “multi-signature” scheme is used, then each
1052 manager can have its own independent secret-public key pair, enabling an [auditable](#) mode
1053 where it possible to check which managers participated, thereby facilitating accountability.

1054 **A.6 Distribution of trust across secure environments**

1055 Hardware security modules (HSMs) are often used to safeguard high-value secret keys.
1056 They perform cryptographic operations, such as signatures, only inside a hardened-security
1057 environment that attempts to prevent exfiltration of the keys. However, even HSMs are
1058 subject to new vulnerabilities and side-channel attacks that enable an insider attacker, with
1059 physical access to an HSM, to exfiltrate a signing key before the HSM is patched. To mitigate
1060 this attack, it is possible to use a diversity of HSMs as multiple parties in a threshold scheme.

1061 For certain threshold schemes, such as for a threshold RSA signature, each HSM only
1062 has to perform an already supported cryptographic operation. Each HSM simply computes
1063 and outputs a regular RSA signature, using a signing key share, and then some external non-
1064 HSM device combines the output shares to obtain the final RSA signature. This application
1065 can be enabled by a dealer that, in an initial safe/protected phase, secret-shares the RSA
1066 key, and distributes one share to each HSM (across diverse locations). For more complex
1067 threshold schemes (including RSA key generation without a dealer), the threshold operations

1068 may require customized programming and interactions between parties. This can be achieved
1069 for example by diverse virtual machines running in various and diversified computers (e.g.,
1070 with different operating systems and protected by different access control mechanisms).

1071 **A.7 Distributed password authentication**

1072 In a typical password-based authentication, a client sends its username and password to a
1073 server, via an encrypted channel, and then the server computes a salted hash of the password
1074 and checks the result against a verification table of hashes. This setting has several single-
1075 points of failure: (i) if the server fails, then the authentication service becomes unavailable;
1076 (ii) if the server's database is leaked by an intruder, then an attacker can use an offline
1077 "dictionary attack" to find which passwords in the dictionary match the database; also, (iii)
1078 if the server is hacked with spyware, then the intruder may be able to read in real time the
1079 passwords sent by clients.

1080 Without changing the underlying hash-based mechanism, the first two mentioned issues
1081 can be rectified by a simple threshold approach. Each salt in the verification table can be
1082 secret-shared across a set of n servers, such that any subset of f or fewer shares has no
1083 information about the not-in-use verification salts, and any subset of $f + 1 + a$ uncompro-
1084 mised servers (for some non-negative a) can reconstruct a verification (salted) hash when so
1085 requested. In this example, the enhanced confidentiality of the values stems from the thresh-
1086 old property of the threshold secret sharing, without using any encryption. The use of salts
1087 prevents the attacker from benefiting from pre-computations in the actual case where the
1088 verification table is leaked (if more than the threshold number of servers is compromised).

1089 The online attack (issue iii above) can be addressed with extra steps, such as for example:
1090 (i) the client sends the password in a **shared-I** mode, i.e., as separate secret shares to each
1091 server; (ii) then the servers, each also with a salt share, jointly compute the salted hash, but
1092 without even recombining the salt (efficiency-wise this may benefit from a hash function
1093 that is friendly with respect to distributed computations); (iii) if the output matches the
1094 expected hash, then the user is authenticated. Thus, besides the secret-sharing of the input,
1095 the complexity of the operation lies only on the side of the servers.

1096 The above description is meant for illustration purposes only. An actual consideration for
1097 a real authentication scheme with threshold properties would require a proper security anal-
1098 ysis and would likely warrant further considerations. For example, other solutions exist to
1099 prevent the client from leaking any information about the password. Some of these solutions
1100 are implemented in practice in the space of password authenticated key exchange (PAKE),
1101 and their threshold variants could be performed using threshold versions of oblivious PRFs.
1102 These can be resilient against an active eavesdropper even if the client does not have an
1103 initial secure channel with the servers. However, some of these solutions go beyond the
1104 scope of the threshold modes currently defined in Section 2.3, since they require the client
1105 to actively participate in a secure computation, performing actions beyond secret sharing.