

# **NIST Internal Report**

## **NIST IR 8214C ipd**

### **NIST First Call for Multi-Party Threshold Schemes**

#### **(Initial Public Draft)**

Luís T. A. N. Brandão  
René Peralta

This publication is available free of charge from:  
<https://doi.org/10.6028/NIST.IR.8214C.ipd>

10 **NIST Internal Report**  
11 **NIST IR 8214C ipd**

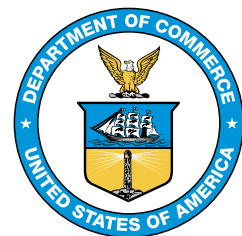
12 **NIST First Call for Multi-Party Threshold Schemes**  
13 **(Initial Public Draft)**

14 Luís T. A. N. Brandão\*  
15 *Strativia*

16 René Peralta  
17 *Computer Security Division*  
18 *Information Technology Laboratory*

19 This publication is available free of charge from:  
20 <https://doi.org/10.6028/NIST.IR.8214C.ipd>

21 January 2023



23 U.S. Department of Commerce  
24 *Gina M. Raimondo, Secretary*

25 National Institute of Standards and Technology  
26 *Laurie E. Locascio, NIST Director and Under Secretary of Commerce for Standards and Technology*

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

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

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

#### **NIST Technical Series Policies**

[Copyright, Fair Use, and Licensing Statements](#)  
[NIST Technical Series Publication Identifier Syntax](#)

#### **Publication History**

This version is the initial public draft (ipd).

#### **How to cite this NIST Technical Series Publication**

Luís T. A. N. Brandão, René Peralta (2023). NIST First Call for Multi-Party Threshold Schemes (Initial Public Draft). (National Institute of Standards and Technology, Gaithersburg, MD) NIST IR 8214C ipd.  
<https://doi.org/10.6028/NIST.IR.8214C.ipd>

#### **NIST Author ORCID identifiers**

Luís T. A. N. Brandão: [0000-0002-4501-089X](#)  
René Peralta: [0000-0002-2318-7563](#)

#### **Contact Information**

[nistir-8214C-comments@nist.gov](mailto:nistir-8214C-comments@nist.gov)

#### **Public Comment Period**

January 25, 2023 – April 10, 2023

#### **Submit Comments**

Only via email: [nistir-8214C-comments@nist.gov](mailto:nistir-8214C-comments@nist.gov)

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

## 59 Reports on Computer Systems Technology

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

## 68 Abstract

69 This document calls for public submissions of multi-party threshold schemes, to support the  
70 National Institute of Standards and Technology (NIST) in developing future recommenda-  
71 tions and guidelines. In a threshold scheme, an underlying key-based cryptographic primitive  
72 is executed while a private/secret key is or becomes secret-shared across various parties.  
73 Submissions in response to this call should include security characterization, technical  
74 description, open-source implementation, and performance evaluation. Submitted threshold  
75 schemes should produce outputs that are “interchangeable” with a key-based cryptographic  
76 primitive of interest. There are two **categories** of primitives for the submission of threshold  
77 schemes: Cat1, for selected NIST-specified primitives; and Cat2, for primitives not specified  
78 by NIST, but which are *friendlier* (more amenable to) to the threshold paradigm, have  
79 enhanced functional features, or/and are based on different cryptographic assumptions. The  
80 analysis of Cat1-submissions will help develop future recommendations and guidelines for  
81 threshold implementations of the corresponding NIST-specified primitives. The analysis of  
82 Cat2-submissions will help assess new interests on primitives not standardized by NIST.

## 83 Keywords

84 Cryptography; distributed systems; provable security; secure multi-party computation;  
85 standards; threshold cryptography; threshold schemes.

## 86 Preface

### 87 Please do not yet submit any threshold scheme.

88 The present **draft** is published for the purpose of obtaining public feedback. The final version  
89 of the “NIST First Call for Multi-Party Threshold Schemes” will consider received feedback  
90 about this document and will integrate other formal components. Please submit feedback  
91 comments to [nistir-8214C-comments@nist.gov](mailto:nistir-8214C-comments@nist.gov) by April 10, 2023.

92 This document is intended for: technicians engaged in the development of recommendations  
93 for threshold schemes; cryptography experts interested in providing constructive technical  
94 feedback, or in collaborating in the development of open reference material; and all those,  
95 including from academia, industry, government and the public in general, interested in future  
96 recommendations about threshold schemes. Relevant preliminary context about this call  
97 can be found in the [NIST-IR8214A](#) (2020), the [MPTC-Call2021a](#) for feedback on criteria for  
98 threshold schemes (2021), and the [NIST-IR8214B-ipd](#) (2022).

## 99 Acknowledgments

100 The first author performed this work as a Foreign Guest Researcher (non-employee) at  
101 NIST, while under a contract with (employed by) Strativia. The authors thank their NIST  
102 colleagues Lily Chen, Michael Davidson, Dustin Moody, Ray Perlner, and Meltem Sönmez  
103 Turan, for their feedback on diverse aspects of this call. The authors also thank Isabel Van  
104 Wyk, from NIST, for various editorial comments.

105 **Call for Patent Claims**

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

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

- 114 a) assurance in the form of a general disclaimer to the effect that such party does not  
115 hold and does not currently intend holding any essential patent claim(s); or
- 116 b) assurance that a license to such essential patent claim(s) will be made available  
117 to applicants desiring to utilize the license for the purpose of complying with the  
118 guidance or requirements in this ITL draft publication either:
  - 119 i) under reasonable terms and conditions that are demonstrably free of any unfair  
120 discrimination; or
  - 121 ii) without compensation and under reasonable terms and conditions that are dem-  
122 onstrably free of any unfair discrimination.

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

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

130 Such statements should be addressed to: [nistir-8214C-comments@nist.gov](mailto:nistir-8214C-comments@nist.gov)

131 **Table of Contents**

132	1. Introduction . . . . .	1
133	2. Acronyms . . . . .	3
134	3. Call and Scope for Submissions . . . . .	6
135	3.1. Category 1 (Cat1) . . . . .	6
136	3.2. Category 2 (Cat2) . . . . .	7
137	3.3. Vision . . . . .	8
138	4. Components of a Submission . . . . .	10
139	4.1. Phases Until Full Submission . . . . .	10
140	4.2. Main component M1: Written specification . . . . .	11
141	4.2.1. Frontmatter . . . . .	11
142	4.2.2. Main matter . . . . .	12
143	4.2.3. Backmatter . . . . .	13
144	4.3. Main component M2: Reference Implementation . . . . .	14
145	4.4. Main component M3: Execution Instructions . . . . .	15
146	4.5. Main component M4: Experimental evaluation . . . . .	16
147	4.5.1. Experimental setting . . . . .	16
148	4.5.2. Measurements . . . . .	16
149	4.5.3. Analysis . . . . .	17
150	4.6. Main component M5: Additional Statements . . . . .	17
151	5. Technical Requirements (T) for Submission of Threshold Schemes . . . . .	18
152	5.1. T1: Primitives . . . . .	18
153	5.2. T2: System Model . . . . .	18
154	5.2.1. T2.1: Participants . . . . .	19
155	5.2.2. T2.2: Distributed Systems and Communication . . . . .	20
156	5.2.3. T2.3: Adversary . . . . .	20

157	5.3. T3: Security Idealization . . . . .	21
158	5.4. T4: Security Versus Adversaries . . . . .	22
159	5.4.1. T4.1: Active Security (Against Active Corruptions) . . . . .	22
160	5.4.2. T4.2: Adaptive Security (Against Adaptive Corruptions) . . . . .	22
161	5.4.3. T4.3: Proactive Security (Against Mobile Attacks) . . . . .	23
162	5.5. T5: Threshold Profiles . . . . .	23
163	5.6. T6: Building Blocks . . . . .	25
164	6. Cat1 primitives — Specified by NIST . . . . .	26
165	6.1. Input/Output (I/O) Interfaces . . . . .	27
166	6.2. Cryptographic Parameters . . . . .	27
167	6.2.1. Elliptic Curves, for ECC-related Primitives . . . . .	27
168	6.2.2. RSA Modulus, for RSA-related Primitives . . . . .	29
169	7. Cat2 Primitives — Not Specified by NIST . . . . .	30
170	7.1. “Regular” Primitives (Subcategories C2.1–C2.5) . . . . .	30
171	7.2. “Other” Primitives/Schemes (Subcategories C2.6–C2.8) . . . . .	31
172	7.2.1. Cat2 subcategory C2.6: “Advanced” . . . . .	31
173	7.2.2. Cat2 subcategory C2.7: ZKPoK . . . . .	31
174	7.2.3. Cat2 subcategory C2.8: Auxiliary Gadgets . . . . .	32
175	A. Details for Subcategories and Primitives of Interest . . . . .	33
176	A.1. Subcategory C1.1: Cat1 Signing . . . . .	33
177	A.1.1. Subcategory C1.1.1: EdDSA Signing . . . . .	33
178	A.1.2. Subcategory C1.1.2: ECDSA Signing . . . . .	34
179	A.1.3. Subcategory C1.1.3: RSADSA Signing . . . . .	35
180	A.1.4. Signing in Secret-Shared-Input (SSI) Mode . . . . .	35
181	A.2. Subcategory C1.2: Cat1 Public-Key Encryption (PKE) . . . . .	35
182	A.2.1. Subcategory C1.2.1: RSA Encryption (of a Secret-Value) . . . . .	36
183	A.2.2. Subcategory C1.2.2: RSA Decryption . . . . .	37
184	A.2.3. Implementation Recommendations and Options . . . . .	37



185	A.3. Subcategory C1.3: Cat1 ECC Primitives for Pair-Wise Key-Agreement (2KA)	39
186	A.3.1. Subcategory C1.3.1: ECC-CDH Primitive . . . . .	42
187	A.3.2. Subcategory C1.3.2: ECC-MQV Primitive . . . . .	42
188	A.4. Subcategory C1.4: Cat1 “Symmetric” . . . . .	43
189	A.4.1. Subcategory C1.4.1: AES Enciphering/Deciphering . . . . .	43
190	A.4.2. Subcategory C1.4.2: KDM and KC for 2KE . . . . .	44
191	A.4.2.1. Key Derivation Mechanism (KDM) . . . . .	45
192	A.4.2.2. Key Confirmation (KC) . . . . .	45
193	A.5. Subcategory C1.5: key-Generation (keygen) for Cat1 Schemes . . . . .	45
194	A.5.1. Subcategory C1.5.1: ECC Keygen (for ECDSA, EdDSA, and ECC-2KA)	47
195	A.5.2. Subcategory C1.5.2: RSA Keygen . . . . .	47
196	A.5.2.1. Criteria for the RSA Modulus and Primes . . . . .	48
197	A.5.2.2. Criteria for the Private Exponent . . . . .	49
198	A.5.3. Subcategory C1.5.3: Bitstring Keygen . . . . .	49
199	A.6. Subcategory C2.6: Advanced . . . . .	49
200	A.6.1. Use-Case Example: Non-Threshold FHE-Based AES Oblivious Enciphering	50
201	A.6.2. Threshold Schemes for FHE-based AES Oblivious Enciphering . . .	51
202	A.7. Subcategory C2.7: ZKPoKs . . . . .	52
203	A.8. Subcategory C2.8: (Auxiliary) Gadgets . . . . .	55
204	B. Submission Checklists . . . . .	56
205	B.1. Checklist for Submission <b>Ph</b> ases (Ph) . . . . .	56
206	B.2. Checklist for Package <b>Main</b> Components (M) . . . . .	56
207	B.3. Checklist for M1: Written Specification <b>Sections</b> (S) . . . . .	56
208	B.4. Checklist for M2: Open <b>source</b> (Src) Reference Implementation . . . . .	57
209	B.5. Checklist for M3: <b>Execution</b> Instructions (X) . . . . .	57
210	B.6. Checklist for M4: <b>Performance</b> Analysis (Perf) . . . . .	57
211	B.7. Checklist for <b>Technical</b> Requirements (T) . . . . .	57
212	References . . . . .	58

213 **List of Tables**

214	Table 1.	Subcategories of interest in Cat1 . . . . .	6
215	Table 2.	Examples of primitives in subcategories of Cat2 . . . . .	8
216	Table 3.	Labels for some template threshold profiles . . . . .	24
217	Table 4.	Primitives of interest in subcategories of Cat1 . . . . .	26
218	Table 5.	Recommended implementation parameters for Cat1 primitives . . . . .	28
219	Table 6.	Notation of EdDSA versus ECDSA (in Draft FIPS 186-5) . . . . .	34
220	Table 7.	RSA-based primitives per party per RSA-2KE scheme . . . . .	38
221	Table 8.	Seven ECC-2KA schemes . . . . .	40
222	Table 9.	ECC-2KA primitives of interest for thresholdization . . . . .	41
223	Table 10.	Examples of keygen purposes . . . . .	46
224	Table 11.	Criteria for the random primes of an RSA modulus . . . . .	48
225	Table 12.	Example ZKPoKs of interest related to Cat1 primitives . . . . .	53

## 226 1. Introduction

227 Over several decades, the National Institute of Standards and Technology (NIST) has  
228 standardized important key-based cryptographic schemes, in various Federal Information  
229 Processing Standards (FIPS) publications, and in Special Publications in Computer Security  
230 (the SP800 series). For example, they provide specifications for digital signatures [FIPS-  
231 186-5-Draft], public-key encryption [SP800-56B-Rev2], pair-wise key-agreement (including  
232 key-derivation primitives) [SP800-56A-Rev3], and symmetric-key enciphering [FIPS-197].

233 In a traditional description or implementation of a key-based cryptographic primitive, the  
234 operation is performed by an individual party that has access to the private/secret key, when  
235 said key is created (in key-generation) or/and used as input (e.g., for signing, enciphering,  
236 or decryption) in the underlying basic primitives. In a corresponding conventional imple-  
237 mentation, said party is a *single-point of failure* for confidentiality, integrity and availability.

238 Modern cryptography enables a multi-party implementation paradigm, based on devel-  
239 opments in the fields of threshold cryptography, secure **multi-party** computation (MPC)  
240 and distributed systems. In a (multi-party) *threshold* scheme, multiple parties perform a  
241 distributed computation, emulating the operation of a key-based cryptographic algorithm,  
242 without combining the private/secret key in any single place, and ensuring security as long  
243 as the number of corrupted parties does not exceed a certain *threshold*. This enables decen-  
244 tralization of trust regarding the creation, storage and use of the private/secret keys. This  
245 threshold paradigm can be applied to NIST-specified primitives and beyond.

246 The development of recommendations and guidelines for threshold schemes, tapping into  
247 the domain of advanced cryptography, is an important step in addressing various challenges  
248 in cybersecurity and privacy. As part of such development, it is expected that the present  
249 “Call for Multi-Party Threshold Schemes” will motivate broad community engagement for a  
250 diverse set of submissions, followed by expert public scrutiny by stakeholders.

251 Recent context leading to the formulation of this call can be found in the Multi-Party  
252 Threshold Cryptography (MPTC) [project webpage](#), the [NIST-IR8214A](#) (2020) with con-  
253 siderations toward criteria, the [MPTC-Call2021a](#) for feedback on criteria for multi-party  
254 threshold schemes (MPTS), the 2020 [MPTS workshop](#) webpage, and the [NIST-IR8214B-ipd](#)  
255 on threshold EdDSA/Schnorr signatures (2022). The present call has the following goals:

- 256 1. **[Reference material]** Create a basis of properly motivated, specified, implemented  
257 and analyzed threshold schemes, to support future recommendations and guidelines.
- 258 2. **[Threshold feasibility]** Assess the viability of threshold implementations of various  
259 primitives of interest, including of selected NIST-specified primitives.
- 260 3. **[Pertinence of other primitives]** In the threshold context, facilitate an initial assess-  
261 ment of the merits of other cryptographic primitives that may be mature for adoption.

4. **[Quantum resistance and other features]** Help explore the space of threshold readiness in terms of quantum-resistance versus other advanced functional features.

The process of collecting high-quality security formulations, technical descriptions, open implementations, and performance evaluations is intended to compose a body of reference material. This will support a phase of analysis to identify sound approaches, best practices, and reusable building blocks. The results will help shape recommendations and guidelines.

**Two categories for submissions.** To assess the viability of threshold schemes for cryptographic primitives, the present call is organized into two categories of submissions, with regard to the primitives in consideration for thresholdization:

- **Cat1:** Selected NIST-specified primitives used in digital signature schemes in [FIPS-186-5-Draft](#), public-key encryption and respective decryption in [SP800-56B-Rev2](#), elliptic-curve based pair-wise key-agreement in [SP800-56A-Rev3](#), symmetric enciphering/deciphering in [FIPS-197](#), key-derivation and key-confirmation mechanisms in the SP 800-56 series (parts [A](#), [B](#), and [C](#)); and the corresponding key-generations.
- **Cat2:** Primitives not specified by NIST, including primitives for “regular” schemes of type similar to those in Cat1 (signing, public-key encryption, key-agreement, enciphering/deciphering, key-derivation and key-confirmation, and their keygen), primitives for “advanced” functionalities (e.g., fully-homomorphic, identity-based or attribute-based encryption), zero-knowledge proofs/arguments of knowledge (e.g., of a secret-shared private key that is consistent with a public key); and other threshold-auxiliary gadgets. Primitives submitted in Cat2 should aim for threshold-friendliness and may be based on cryptographic assumptions different from those in Cat1. There is a particular interest in combined threshold-friendliness and quantum resistance.

The analysis in Cat1 will help assess threshold friendliness and develop future recommendations and guidelines for threshold schemes of NIST-specified primitives. The analysis in Cat2 will help assess new interests on primitives not currently standardized by NIST, and help characterize the possible alignment between (i) threshold-friendliness, (ii) quantum resistance, and (iii) additional useful features. This may also serve as relevant input to assess the ability to deploy secure multi-party applications with advanced privacy features.

**Organization.** Section [2](#) explains the acronyms used in the document. Section [3](#) calls for submissions and explains the partition into two categories. Section [4](#) enumerates logistic and formatting requirements for the submission of packages. Section [5](#) defines technical requirements for threshold schemes. Section [6](#) lists primitives and threshold modes of interest for each subcategory of Cat1 (NIST-specified primitives), mentioning possible I/O interfaces and recommending cryptographic parameters. Section [7](#) describes the subcategories of interest in Cat2 (primitives not specified by NIST). Appendix [A](#) provides further details about subcategories. Appendix [B](#) displays a checklist of the elements of a submission.

## 299 2. Acronyms

300	Acronym	Extended form
301	2KA	Pair-wise <b>key</b> -agreement
302	2KE	Pair-wise <b>key</b> -establishment
303	ABE	Attribute- <b>b</b> ased <b>E</b> ncryption
304	AEAD	Authenticated encryption with <b>a</b> ssociated <b>d</b> ata
305	AES	Advanced <b>E</b> ncryption <b>S</b> tandard
306	API	Application <b>p</b> rogramming <b>i</b> nterface
307	CDH	Cofactor <b>D</b> iffie- <b>H</b> ellman
308	CMAC	Cipher-based <b>M</b> AC
309	CPU	Central <b>p</b> rocessing <b>u</b> nit
310	CRS	Common <b>r</b> eference string
311	CRT	Chinese <b>r</b> emainder theorem
312	DKG	Distributed <b>k</b> ey <b>g</b> eneration
313	DOI	Digital <b>o</b> bject <b>i</b> dentifier
314	ECC	Elliptic <b>c</b> urve cryptography
315	ECDSA	Elliptic Curve <b>D</b> igital <b>S</b> ignature Algorithm
316	EdDSA	Edwards Curve <b>D</b> igital <b>S</b> ignature Algorithm
317	FFC	Finite <b>f</b> ield <b>c</b> ryptography
318	FHE	Fully- <b>h</b> omomorphic encryption
319	FIPS	Federal <b>I</b> nformation <b>P</b> rocessing Standards
320	FR	Field <b>r</b> epresentation indicator
321	GB	<b>G</b> igabyte (1,000,000,000 bytes)
322	GC	<b>G</b> arbled circuit
323	HMAC	<b>H</b> ash-based <b>M</b> AC
324	IBE	Identity- <b>b</b> ased encryption
325	IETF	Internet <b>E</b> ngineering <b>T</b> ask <b>F</b> orce
326	I/O	Input/output
327	IRTF	Internet <b>R</b> esearch <b>T</b> ask <b>F</b> orce
328	ITL	Information <b>T</b> echnology <b>L</b> aboratory

	<b>Acronym</b>	<b>Extended form</b>
300		
329	KA	<b>K</b> ey <b>a</b> greement
330	KAS1/2	<b>K</b> ey <b>a</b> greement scheme 1 or 2
331	KAT	<b>K</b> nown- <b>a</b> nswer <b>t</b> est
332	KC	<b>K</b> ey <b>c</b> onfirmation
333	KDM	<b>K</b> ey- <b>d</b> erivation <b>m</b> echanism
334	KT	<b>K</b> ey- <b>t</b> ransport
335	KMAC	<b>K</b> eccak-based <b>M</b> AC
336	LCM	<b>L</b> east <b>c</b> ommon <b>m</b> ultiplier
337	LTS	<b>L</b> ong <b>t</b> erm support
338	LWC	<b>L</b> ightweight <b>C</b> ryptography
339	MAC	<b>M</b> essage <b>a</b> uthentication <b>c</b> ode
340	MPC	(Secure) <b>m</b> ultiparty <b>c</b> omputation
341	MPTC	<b>M</b> ulti- <b>P</b> arty <b>T</b> hreshold <b>C</b> ryptography
342	MPKA	<b>M</b> ultiparty <b>k</b> ey <b>a</b> greement
343	MQV	<b>M</b> enezes- <b>Q</b> u- <b>V</b> anstone
344	NIST	<b>N</b> ational <b>I</b> nstitute of <b>S</b> tandards and <b>T</b> echnology
345	NIZK	<b>N</b> on-interactive <b>z</b> ero- <b>k</b> nowledge
346	NISTIR	<b>NIST</b> <b>I</b> nternal <b>R</b> eport
347	NSS	<b>n</b> ot- <b>s</b> ecret- <b>s</b> hared (input/output)
348	OAEP	<b>O</b> ptimal <b>A</b> symmetric <b>E</b> ncryption <b>P</b> adding
349	PC	<b>P</b> ersonal <b>c</b> omputer
350	PDF	<b>P</b> ortable <b>d</b> ocument <b>f</b> ormat
351	PF	<b>P</b> latform
352	PEC	<b>P</b> rivacy- <b>E</b> nhancing <b>C</b> ryptography
353	PQC	<b>P</b> ost- <b>Q</b> uantum <b>C</b> ryptography
354	PKC, PKCS	<b>P</b> ublic- <b>K</b> ey <b>C</b> ryptography, <b>PKC</b> Standards
355	PKE	<b>P</b> ublic- <b>k</b> ey <b>e</b> ncryption
356	PRF	<b>P</b> seudorandom <b>f</b> unction family
357	PRP	<b>P</b> seudorandom <b>p</b> ermutation family

	<b>Acronym</b>	<b>Extended form</b>
300		
358	PSS	<b>P</b> robabilistic signature scheme
359	PVSS	<b>P</b> ublicly verifiable secret sharing
360	QR	<b>Q</b> uantum- <b>r</b> esistant or <b>q</b> uantum <b>r</b> esistance
361	RAM	<b>R</b> andom access <b>m</b> emory
362	RBG	<b>R</b> andom- <b>b</b> it generator/generation
363	RFC	<b>R</b> equ <del>e</del> st for <b>C</b> omments
364	RO	<b>R</b> andom <b>o</b> racle
365	RSA	<b>R</b> ivest– <b>S</b> hamir– <b>A</b> dleman
366	RSADP	<b>RSA</b> <b>D</b> ecryption <b>P</b> rimitive
367	RSADSA	<b>RSA</b> <b>D</b> igital Signature <b>A</b> lgorithm
368	RSAEP	<b>RSA</b> <b>E</b> ncryption <b>P</b> rimitive
369	RSASSA	<b>RSA</b> Signature Scheme with <b>A</b> ppendix
370	RSASVE	<b>RSA</b> Secret- <b>V</b> alue <b>E</b> ncapsulation
371	S2PC	Secure <b>two</b> - <b>p</b> arty computation
372	SHA	Secure <b>h</b> ash <b>a</b> lgorithm
373	SHAKE	Secure <b>h</b> ash <b>a</b> lgorithm with <b>KECCAK</b>
374	SNARK	Succinct <b>n</b> on-interactive <b>a</b> rgument of <b>k</b> nowledge
375	SP 800	Special <b>P</b> ublication in Computer security
376	SSD	Solid state <b>d</b> rive
377	SSI, SSIO	Secret-shared input, secret-shared <b>i</b> nput-and- <b>o</b> utput
378	SSO	Secret-shared <b>o</b> utput
379	SVE	Secret-value <b>e</b> ncapsulation
380	TB	<b>T</b> erabyte (1,000,000,000,000 bytes)
381	TF	<b>T</b> hreshold- <b>f</b> riendly
382	URL	<b>U</b> niform <b>r</b> esource <b>l</b> ocator
383	VSS	<b>V</b> erifiable secret sharing
384	XOF	Extendable <b>o</b> utput <b>f</b> unction
385	ZKP	<b>Z</b> ero <b>k</b> nowledge <b>p</b> roof
386	ZKPoK	<b>Z</b> ero <b>k</b> nowledge <b>p</b> roof of <b>k</b> nowledge

### 3. Call and Scope for Submissions

This document is a **call** for multi-party threshold schemes. It solicits high-quality specifications of threshold schemes for primitives across two **categories**: **Cat1** (selected NIST-specified primitives) and **Cat2** (primitives not specified by NIST). Each submission should include a security characterization, a technical description, an open-source reference implementation, and a performance evaluation. Submitted schemes will benefit from exposure to public analysis, and will be considered in a future report. This is a preliminary phase for collection of reference material, and assessment of threshold schemes. The results of this phase will inform future development of recommendations, and may be considered in possible future efforts for development of guidelines or standards.

#### 3.1. Category 1 (Cat1)

Cat1 consists of selected, stateless, NIST-specified cryptographic primitives, organized in Table 1 across five subcategories:

- **C1.1**, for EdDSA, ECDSA and RSADSA signing [FIPS-186-5-Draft];
- **C1.2**, for RSA encryption (for key-encapsulation) and decryption [SP800-56B-Rev2];
- **C1.3**, for ECC-based pair-wise **key-agreement** (2KA) [SP800-56A-Rev3] via CDH or MQV;
- **C1.4**, for AES-enciphering/deciphering [FIPS-197], and **key-derivation** (KD) and **key-confirmation** (KC) for 2KE [SP800-56C-Rev2; SP800-135-Rev1; SP800-108-Rev1];
- **C1.5**, for ECC keygen [FIPS-186-5-Draft; SP800-56A-Rev3; SP800-186-Draft], RSA keygen [FIPS-186-5-Draft; SP800-56B-Rev2], and bitstring (or integer) keygen.

**Table 1.** Subcategories of interest in Cat1

Subcategory: Type	Families of specifications	Section in this call
<b>C1.1: Signing</b>	EdDSA sign, ECDSA sign, RSADSA sign	<b>A.1</b>
<b>C1.2: PKE</b>	RSA encryption, RSA decryption	<b>A.2</b>
<b>C1.3: 2KA</b>	ECC-CDH, ECC-MQV	<b>A.3</b>
<b>C1.4: Symmetric</b>	AES encipher/decipher, KDM/KC (to support 2KE)	<b>A.4</b>
<b>C1.5: Keygen</b>	ECC keygen, RSA keygen, bitstring keygen	<b>A.5</b>

Note: In the second column, each item within a subcategory is itself called a family of specifications, since it may include diverse primitives or modes/variants, some of which are mentioned in Table 4 (in Section 6).



Section 6 presents more details about versions and modes of primitives in Cat1, including options for input/output interfaces (Section 6.1) and cryptographic parameters recommended for evaluation (Section 6.2). The analysis of Cat1 submissions will facilitate the development of recommendations and guidelines on threshold schemes for the corresponding NIST-specified primitives, highlighting reference approaches, techniques, building blocks, and best practices. The results will be reported in a NIST publication.

### 3.2. Category 2 (Cat2)

The goal of Cat2 is to enable submissions that make a strong case for certain threshold-feasible primitives that are not standardized by NIST. While the scope is wide, Cat2-submissions should be justified on the basis of the primitives being thresholdized having/enabling useful differentiating features, such as having/being: (i) threshold-friendly(ier) (TF); (ii) based on alternative cryptographic assumptions (e.g., pairings), possibly quantum-resistant (QR) (e.g., lattice-based); (iii) useful probabilistic properties (e.g., determinism versus non-determinism), (iv) more efficient in a relevant metric, or/and (v) advanced functional features (e.g., allowing homomorphic computation over encrypted data).

Cat2 has eight subcategories, including five “regular” (somewhat matching the subcategories of Cat1), and three others (“advanced”, “ZKPoK” and “gadgets”), as listed in Table 2:

- “Regular”:

- C2.1, for signing (e.g., verifiably-deterministic succinct signatures, and/or TF-QR);
- C2.2, for PKE (e.g., TF-QR decryption and key-encryption);
- C2.3, for key agreement (e.g., TF primitives that are QR and/or that facilitate low-round key-agreement for more than two parties);
- C2.4, for symmetric-key primitives (e.g., TF enciphering/deciphering), and hashing-related primitives for key derivation and key confirmation;
- C2.5, for keygen for primitives in other subcategories.

- “Others”:

- C2.6, for primitives for cryptographic schemes with **advanced** functional features, e.g., fully-homomorphic, identity-based, and attribute-based encryption schemes.
- C2.7, for zero-knowledge proofs of knowledge (ZKPoK) that are deemed useful to support the threshold setting, such as for proving knowledge of private/secret information consistent with a correct secret-sharing setup.
- C2.8, for other auxiliary “gadgets” deemed useful to support the threshold setting, namely to support the implementation of other threshold schemes in scope.

**Table 2.** Examples of primitives in subcategories of Cat2

Subcategory: Type	Example scheme	Example primitive
C2.1: Signing	Succinct & verifiably-deterministic signatures	Signing
C2.2: PKE	TF-QR public-key encryption (PKE)	Decryption/encryption
C2.3: KA	Low-round multi-party key-agreement (KA)	Single-party primitives
C2.4: Symmetric	TF-QR blockcipher/PRP	Encipher/decipher
	TF-QR key-derivation / key-confirmation	PRF and hash function
C2.5: Keygen	Any of the above	Keygen
C2.6: Advanced	QR fully-homomorphic encryption	Decryption; Keygen
	Identity-based and attribute-based encryption	Decryption; Keygens
C2.7: ZKPoK	ZKPoK of private key	ZKPoK.Generate
C2.8: Gadgets	Garbled circuit (GC)	GC.generate; GC.evaluate

Legend: PRF = pseudorandom function [family]. PRP = pseudorandom permutation [family]. QR = quantum resistant. TF = threshold-friendly. ZKPoK = zero knowledge proof of knowledge.

Section 7 contains more details and examples on Cat2. Some Cat2-submissions may be evaluated within the scope of the NIST Privacy-Enhancing Cryptography (PEC) project [Proj-PEC]. It is expected that the results of this exercise will be reported in a NIST publication.

### 3.3. Vision

**Quantum-resistant versus quantum-breakable primitives.** There is a strong interest in receiving submissions of threshold schemes for threshold-friendly quantum-resistant (TF-QR) primitives. As there is currently a gap between some known useful cryptographic features and quantum-resistance, there is also interest in submissions that have enhanced functional features even if they are only secure with respect to non-quantum adversaries.

**Interchangeability.** This call is scoped on threshold schemes whose output can be used in subsequent operations (e.g., signature verification) that were specified to use the output of the corresponding conventional (non-threshold) primitive (e.g., signing). The intended notion is that of *interchangeability*, from §2.4 of NIST-IR8214A. EdDSA signing provides a notable example: the threshold setting favors a consideration not only of pseudorandom signatures, but also of probabilistic ones that are *interchangeable* in the sense of being verifiable by the standardized EdDSA verification (see NIST-IR8214B-ipd). In Cat1, the primitives of interest are already fixed. In Cat2-submissions, the primitives of interest need to be specified along with the corresponding threshold schemes.

**Provable security.** The security of submitted threshold schemes is expected to be assessed based on *multi-party protocol analysis*, which is supported by a large and mature body of knowledge in *provable security*. This is different from the extensive cryptanalysis that would be required in a call for basic primitives based on new cryptographic assumptions. That said, the security of threshold schemes is still recognized as multi-dimensional, depending on security formulation (e.g., which ideal functionalities or security games to choose), implementation (e.g., susceptibility to side-channels), and deployment suitability (e.g., whether security assumptions are appropriate for the deployment environment).

**Diversity.** The domain space of multi-party threshold schemes is considerably wider than that of the primitives (e.g., digital signatures) being thresholdized. Acknowledging this, the present call allows leeway for the submitters to select from a variety of system models, threshold configurations, security formulations, technical approaches, and benchmarking focuses. Thus, the usual criteria for “apples-to-apples” comparison (e.g., number of parties, common programming language, application programming interface, etc.) will not be required in the initial phase. Nonetheless, the submissions are expected to adhere to certain criteria, with respect to both technical documentation (see Section 4) and technical characteristics of the proposed threshold schemes (e.g., needs to include a security formulation against active corruptions — see Section 5). After a review of the system models proposed in the initial set of submissions, a request may be made for submitters to provide new performance evaluation results (e.g., with a particular number of parties and threshold values) based on adjusted parameters to facilitate a comparison across submissions.

**Initial phase.** The initial phase of analysis is expected to take about one year after the submission deadline, and will consider comments from the public. It will also include a workshop for presentation of the submitted threshold schemes. A NIST report will follow. For [Cat1](#), the results will help determine how the development of future recommendations and guidelines may be differentiated per primitive, and whether it will focus on full-fledged threshold schemes, on identifying building blocks and composition techniques, or a hybrid of these. For [Cat2](#), the results will include an initial characterization of the space of submissions to help assess possible interest in a subsequent more-focused analysis.

**Reliance on contributions.** The success of the process will depend on:

- **high-quality submissions** by teams with appropriate expertise, including in the areas of secure multiparty computation and distributed systems;
- **expert public scrutiny**, including assessments of security;
- **comments on pertinence**, by stakeholders of applications of threshold schemes.

## 515 4. Components of a Submission

### 516 4.1. Phases Until Full Submission

517 The submission process is organized with a deadline for package submissions, while also  
518 considering a possible early abstract and preliminary submission, as follows:

519 **Ph1. (Optional) Early abstract:** No later than **about 90 days** (exact date to be deter-  
520 mined) after the final version of this call is published, a short document (with no  
521 more than three pages) can be submitted with a title, a list of team members, and  
522 a preliminary abstract of a planned full package to be submitted later ([Ph3](#)). The  
523 abstract should identify the primitives to be thresholdized and their corresponding  
524 category and subcategory(ies)/type(s), give an outline of the threshold approach  
525 (including system model, the protocol approach, and main security properties), and  
526 list the most relevant bibliographic references. This phase for optional submission  
527 (not mandatory and non-committing) is intended to facilitate early discussion of the  
528 expected coverage of each category/subcategory, and may help determine useful  
529 merges, differentiations, or alternative submissions.

530 **Ph2. (Optional) Preliminary package:** Submission packages received by NIST at least  
531 **45 days before the deadline** for full packages will be early reviewed for complete-  
532 ness. The submitters will be notified of identified deficiencies, tentatively within 25  
533 days, to allow amendments before the deadline.

534 **Ph3. Full package:** Full submission packages must be received by NIST no later than  
535 **about 150 days** (exact date to be determined) after the final version of this call is  
536 published. Despite possible adjustments to be made in this call, submitters are en-  
537 couraged to prepare early for future submissions, using the present draft as a baseline.  
538 A complete and proper package must contain the following **main** components:

- 539 • **M1. Written specification:** A technical specification (including security analy-  
540 sis) of the threshold scheme and primitives (see Section [4.2](#)).
- 541 • **M2. Reference implementation:** An open-source implementation (software),  
542 including code, license, comments, and explaining an API (see Section [4.3](#)).
- 543 • **M3. Execution instructions:** Instructions to enable the execution of the thresh-  
544 old scheme and reproduction of experimental results (see Section [4.4](#)).
- 545 • **M4. Experimental evaluation:** A report describing an experimental setting,  
546 measuring performance, and interpreting the results (see Section [4.5](#)).
- 547 • **M5. Additional statements:** Various statements (see Section [4.6](#)).

548 **Submissions medium.** The submission of any documentation — early abstract (Ph1),  
549 preliminary package (Ph2), full package (Ph3), or any amendment — must be at least  
550 confirmed by sending an email to [MPTS-submissions@nist.gov](mailto:MPTS-submissions@nist.gov). The final version of this  
551 call may specify a complementary platform to help manage the process of submission and  
552 review. More-specific instructions will be provided in the final version of this call.

553 **Public posting.** after the SUBMISSION deadlines, approved submissions of early abstracts  
554 (Ph1) and full packages (Ph3) will be posted online, and hyperlinked from the MPTC project  
555 website [[Proj-MPTC](#)], for public review.

556 **Note on LaTeX templates.** To facilitate some common document structure across submis-  
557 sions, the final version of the call will provide LaTeX-based templates applicable to some of  
558 the submission documents, for compilation into portable document format (PDF) files.

559 **Note on multiple threshold schemes per package.** A submission package may include a  
560 family of distinguished threshold schemes based on common building blocks, and whose  
561 implementations may make use of common portions of open-source code. Even if a  
562 submission package proposes more than one threshold scheme, each of the above-mentioned  
563 five components should appear only once, possibly using subsections (when applicable) to  
564 distinguish which primitives/schemes the comments relate to.

## 565 4.2. Main component M1: Written specification

566 Submitted specifications of threshold schemes must be compiled in a PDF document,  
567 written in English and aided with mathematical notation, containing various (numbered or  
568 unnumbered) sections, as described ahead across a frontmatter (see Section 4.2.1), a main  
569 matter (see Section 4.2.2), and backmatter (see Section 4.2.3).

### 570 4.2.1. Frontmatter

571 **S1. Title pages:** Two title-pages, as follows:

- 572 • A first title-page (cover page) with: a title for the proposed submission, the names  
573 and affiliations of the submitters; and the submission date.
- 574 • A second title-page, with all content of the first title-page, and additionally includ-  
575 ing: contact email-addresses for all the submitters; applicable disclaimers related  
576 to affiliations and funding; and, if applicable, other pertinent information about the  
577 team and the submission.

578 **S2. Abstract:** A text with up to 500 words, identifying the primitives being thresholdized,  
579 their corresponding category and subcategory/type in the scope of this call, and the  
580 types of threshold schemes being proposed (i.e., their main features, cryptographic  
581 assumptions and performance highlights).

582 **S3. Executive summary:** An abridged explanation (up to four pages) of the content of  
583 the submission, highlighting relevant properties of the proposed threshold schemes,  
584 their applicability, their performance, and some of the challenges (e.g., in proving  
585 security). It should also briefly mention the submitted components beyond the  
586 specification, including the open-source software with reference implementation.

587 **S4. Index:** A table of contents (i.e., index of sections, subsections, etc.); and (however  
588 applicable) lists of figures, tables, pseudo-code, and other relevant enumerated com-  
589 ponents. Each referenced element in the index should be hyperlinked to the respective  
590 position in the document, and also indicate the corresponding page number.

#### 591 4.2.2. Main matter

592 **S5. Clarification of prior work:** An enumeration of the building blocks, techniques and  
593 ideas known to have been developed or authored in prior work and that are used in  
594 the specification of the primitives and threshold schemes of the present submission.  
595 With regard to the building blocks, techniques and ideas in the submission (preferably  
596 including hyper-references to the related portions of the submitted specification),  
597 this section should aim to clarify and distinguish between (i) those that may have  
598 been designed by authors that are not part of the submitters' team, (ii) those that may  
599 have been previously developed/authored by members of the submitters' team, and  
600 (iii) those that may be original in the present submission. Appropriate bibliographic  
601 references should be given where applicable, preferably including (when possible)  
602 a hyperlink to online-accessible documentation. If applicable, this section can also  
603 include known information pertinent to the "[call for patent claims](#)".

604 **S6. Conventional primitives/scheme:** A review of the conventional (non-threshold)  
605 primitives/scheme that constitute the objects of thresholdization and determine the  
606 interchangeability requirements. For example, if a submitted package proposes a  
607 threshold scheme for ECDSA signing, then this section will provide a brief review  
608 of the conventional ECDSA signing algorithm, and the requirements related to  
609 the corresponding keygen and verification algorithms. The notation used in this  
610 description should be consistent with the one later used to describe the threshold  
611 scheme. [Cat2](#)-submissions are expected to be more thorough in this description.

- 612 **S7. System model:** A thorough description of the system model, including participants,  
613 communication network, and adversary (see [T2](#)).
- 614 **S8. Protocol description:** A detailed description of the multi-party threshold scheme,  
615 modularizing the description of primitives/gadgets where appropriate.
- 616 **S9. Security analysis:** A detailed security analysis, including security formulation (e.g.,  
617 ideal functionalities and/or games), proof(s) of security, and discussion of security  
618 properties and ideal components (see [T3](#) and [T4](#)).
- 619 **S10. Analytic complexity:** An analytical estimation of (i) memory complexity, (ii) com-  
620 putational complexity, (ii) communication complexity, and (iii) round complexity.  
621 The estimates should: include a breakdown across the various possible phases of the  
622 protocol; clarify the complexity per party versus the aggregate in the entire system;  
623 clarify its dependence on various configurable parameters, such as for example the  
624 security strength, the number of parties and the thresholds.
- 625 **S11. Choices and comparisons:** A rationale for design decisions and the chosen system  
626 model, as well as an explanation of known advantages and limitations compared to  
627 other options and approaches.
- 628 **S12. Technical criteria:** An evaluation of various items of technical criteria (see Section [5](#)  
629 and Section [B.7](#)).
- 630 **S13. Deployment recommendations:** A set of deployment requirements and recommen-  
631 dations, including those related to security. This section should also include a list of  
632 known and proposed applications of the submitted threshold scheme(s).
- 633 **4.2.3. Backmatter**
- 634 **S14. Notation:** A section explaining the notation, including:
- 635 • a list of the used acronyms, and their extended expressions;
  - 636 • a list of the used abbreviations, and their complete words;
  - 637 • a list of the used mathematical symbols, and their brief explanations;
  - 638 • (optional) a glossary of selected important terms, with succinct explanations.
- 639 **S15. References:** A list of external references cited throughout the document, ideally  
640 including persistent identifiers (e.g., DOI, and ia.cr) and a link to a corresponding  
641 publicly and (when possible) freely accessible version of the referenced document.



642 **S16. Appendices:** Auxiliary elements deemed too detailed or cumbersome for a first  
643 read may be deferred to appendices, at the end of the document, as long as properly  
644 referenced and hyperlinked in the corresponding above-mentioned sections.

#### 645 **4.3. Main component M2: Reference Implementation**

646 **Required clear implementation.** The submissions packages must contain **open-source**  
647 **code** (software), including explanatory inline comments, constituting a “clear” reference  
648 implementation of the proposed threshold scheme(s). The code and comments should strive  
649 for clarity and understanding, even if at some detriment to efficiency. Optionally, some  
650 modules may include additional code optimized for some efficiency metric(s), to enable  
651 demonstration of better experimental performance.

652 The implementation(s) must support all main features of the threshold scheme and be  
653 suitable to run each “party” in a modern **personal computer** (PC). To facilitate testing, the  
654 implementation should enable “running” the set of all parties in a *baseline* **platform** (PF1)  
655 consisting of a single PC (possibly virtualized), equipped with:

- 656 1. **Processor:** Central processing unit (CPU) with up to eight 64-bit processing cores.
- 657 2. **Fast primary memory:** Up to 32 gigabytes (e.g., of random-access **memory** [RAM])
- 658 3. **Secondary memory:** Up to 4 terabytes (e.g., in a solid state **drive** [SSD])

659 The code (and its instructions) should be designed to allow for a compilation and execution  
660 of the submitted implementation on top of a Linux Ubuntu Desktop 22.04.1 long-term  
661 support (LTS) operating system running installed in platform PF1, without requiring software  
662 download from external sources. Each party should be executed as one (or more) process(es),  
663 or within a software virtual container, separate from the other parties.

664 The submitted open-source software (and documentation) should satisfy the following:

665 **Src1. Is self-contained:** The code was tested to compile and execute properly within the  
666 baseline platform (PF1) with a Linux Ubuntu Desktop v22.04.1 operating system.

667 **Src2. Is licensed as open-source:** The code is explicitly licensed as open-source (e.g.,  
668 possibly based on a license listed in <https://opensource.org/licenses>).

669 **Src3. Contains inline comments:** The code is explained with auxiliary comments.

670 **Src4. Has a clear API:** It explains the **application programming interface** (API), aimed  
671 at facilitating (i) testing, (ii) use in higher-level applications, and (iii) comparison  
672 of performance with other implementations that may follow the same API.



**On programming choices.** As explained in Section 3.3, it is intentional that this call does not specify a concrete programming language, compiler, or API to be used across submissions. That said, it would be useful that the provided open-source reference implementation comes accompanied with explained rationale for choices made. This may include recommendations on the API that future implementations should follow to be easily comparable with the provided reference implementation.

**On validation and verification.** The validation of implementations and formal verification are not included as technical requirements for this call. However, it is expected that the public scrutiny of submitted schemes (namely their specifications and implementations) will facilitate the production of high-assurance software. The analysis of the submissions may clarify what software testing may be proposed across various types of threshold schemes.

#### 4.4. Main component M3: Execution Instructions

A submission package must include execution instructions, as follows:

1. **User manual:** A “user manual” with instructions (and examples) on:

- X1. Compilation:** How to compile the open-source code.

- X2. Parametrization:** How to configure execution parameters, such as the number of parties, the corruption threshold, the type of communication channels, some adversarial choices, and some client choices (e.g., input to the cryptographic primitive). Preferably the configuration of each parameter can be done via the editing of a human-readable text file, and/or command line arguments.

- X3. Execution:** How to test and execute the various phases of the proposed threshold schemes and underlying primitives.

- X4. KAT set:** A set of “known answer-test” (KAT) values, to aid in correctness verification of the execution of the protocol.

2. **Set of scripts:**

- X5. KAT-script:** A script to automatically execute the threshold schemes in a way that reproduces the set of KAT values (X4) provided in the user manual.

- X6. Benchmark-script:** A script to automatically benchmark the threshold scheme in platform PF1, using the “clear” reference implementation, to produce a table recording various performance measurements (similar to that required in Section 4.5) for various configurations. If the submitted implementation

704 includes additional code optimized for performance, and whose performance  
705 results are reported in [M4](#), then corresponding scripts should also be provided,  
706 to enable reproducibility of results.

707 **X7. Other scripts (optional):** Optionally, other scripts to provide better insights  
708 into the workings of the underlying primitives and threshold scheme.

#### 709 **4.5. Main component M4: Experimental evaluation**

710 The package must include a report on experimental performance, obtained by executing the  
711 provided code in the baseline platform (PF1), evaluating a representative set of configurations  
712 supported by the proposed threshold scheme(s). The report must describe:

- 713 1. the experimental setting (see Section [4.5.1](#));
- 714 2. the measured performance (see Section [4.5.2](#)); and
- 715 3. an analysis/interpretation of the results (see Section [4.5.3](#)).

##### 716 **4.5.1. Experimental setting**

717 The report must describe the expected performance characteristics of the experimental setting  
718 (namely of the underlying hardware) supporting the baseline implementation platform PF1.  
719 The description must describe at least the relevant expected characteristics of the (possibly  
720 emulated) processor (e.g., instruction set, and clock frequency), communication network  
721 (e.g., bandwidth, and latency), and memory (e.g., read and write speed).

722 The benchmarking can also include experimentation with different platforms (PF2, ...) of  
723 the submitter's choice (motivated by real or conceivable applications). The performance  
724 results obtained with these alternative platforms (to also be described) may be better or worst  
725 than with PF1. For example, if there are more than eight parties and all require intensive  
726 computing, then the testing in a platform with more than eight cores may provide better  
727 results than with the baseline PF1.

##### 728 **4.5.2. Measurements**

729 The evaluation of experimental performance should report, at least for platform PF1, at least  
730 the following metrics:

- 731 • **Perf1. Memory complexity** (in # bytes required to be simultaneously stored).
- 732 • **Perf2. Processing time** (in seconds) and/or processing (e.g., # of processing cycles).

- 733 • **Perf3. Communication complexity** (in # communicated bytes).
- 734 • **Perf4. Networking time** (in seconds).
- 735 • **Perf5. Round complexity** (in # alternations of the direction of communicated messages).

736 The mentioned metrics should be evaluated and reported in (i) total per execution, (ii) per  
737 identifiable phase of the protocol, and (iii) per party. The results can be reported across  
738 various configurations, e.g., with distinct numbers of parties, and across two distinct security  
739 strengths (e.g., 128 and 224–256 bits).

740 The reported measurements should include results obtained with the submitted “clear”  
741 reference implementation (see Section 4.3). If the submission includes additional code  
742 optimized for performance, then the corresponding results can be added to the measurements’  
743 report. As prescribed in X7, all these benchmarking should be reproducible by a simple  
744 execution of the submission-required scripts.

#### 745 4.5.3. Analysis

746 The performance analysis should include a written explanation/interpretations of the ex-  
747 perimental results, indicating expected or unexpected observations (e.g., some observed  
748 correlation between some complexity metric and the number of parties). The comparison  
749 of results across different configurations and/or experimental settings may be useful to  
750 understand, test or verify tradeoffs and scalability of the system across different metrics.

#### 751 4.6. Main component M5: Additional Statements

752 The packages must include certain statements (on intellectual property, agreements or dis-  
753 closures) to ensure free worldwide availability of the submitted packages for public review  
754 and evaluation purposes, and allowing derivative work and use, in particular for the possi-  
755 ble future elaboration and publication of recommendations, guidelines and standards. The  
756 concrete statements (to be included or referenced in the final version of this call) will be  
757 aligned with the NIST [ITL Patent policy](#), and are likely to be similar to those used by the  
758 NIST Post-Quantum Cryptography (PQC) project [[Proj-PQC](#)].

## 759 5. Technical Requirements (T) for Submission of Threshold Schemes

760 In addition to the structural [requirements for submission packages](#), the specification of  
761 threshold schemes is subject to certain technical requirements (T1–T6) at a logical level.  
762 The following are based on a previous call for feedback on criteria [[MPTC-Call2021a](#)].

### 763 5.1. T1: Primitives

764 A submitted specification must explain in [S6](#) the conventional (non-threshold) primitives  
765 (e.g., decryption) that are the object of thresholdization. Each such primitive must be framed  
766 within the subcategories structure established for Cat1 (see Sections [3.1](#) and [6](#)) and Cat2  
767 (see Sections [3.2](#) and [7](#)). The primitive must also be explained within the scope of an  
768 underlying conventional scheme, composed of various primitives. For example, a decryption  
769 primitive of a **public-key encryption (PKE)** scheme relates to corresponding encryption and  
770 key-generation primitives. The explanation of the primitive must define the corresponding  
771 scope of [interchangeability](#), to be considered by the proposed threshold scheme.

772 Notwithstanding the advantage of referenceability to NIST specifications, a submission  
773 in Cat1 still needs to include a technical description of the primitives being thresholdized.  
774 The description should try to follow the notation and operations specified in the cor-  
775 responding NIST documentation. Some Cat2-submissions may require a more thorough  
776 description, since their underlying non-threshold primitive is not part of a NIST specification.  
777 The explanation should also include references to authoritative descriptions in publicly free  
778 documentation (e.g., papers and standards).

### 779 5.2. T2: System Model

780 A proposal of threshold schemes must strive for a clear description that facilitates under-  
781 standing various options across possible deployment scenarios. Therefore, the specification  
782 of each submitted threshold scheme must describe (in [S7](#)) one system model (and may  
783 identify possible variants), including the set of participants, the communication model and  
784 the adversarial model (goals and capabilities). In addition to the actual “parties” that hold  
785 the secret-shared keys, the system may include coordinators, administrators, clients and  
786 other devices (e.g., routers, clocks, random-bit generators), etc. The model must also explain  
787 how the parties are activated (e.g., via an authorized/authenticated client request, or by an  
788 administrator). See also §2.3 of [NIST-IR8214A](#).

789 Some of the paragraphs ahead describe baseline assumptions and options for a system  
790 model, with regard to participants (Section [5.2.1](#)), communication (Section [5.2.2](#)), and

791 adversary (Section 5.2.3). These assumptions are intended as a baseline, neither precluding  
792 submissions with sophisticated nuances, nor eliminating the utility of security evaluation  
793 across diverse deployment scenarios.

#### 794 5.2.1. T2.1: Participants

795 **The parties in a threshold entity.** There is a “threshold entity” composed on  $n$  “parties”,  
796 responsible for executing a cryptographic primitive. At the onset, all parties “know who” the  
797  $n$  parties are, agreeing on  $n$  identifiers (e.g., possibly public keys to support authenticated  
798 channels). The suitability of public keys may need to be verified, locally or interactively,  
799 possibly via zero-knowledge proofs, in the keygen phase or in subsequent proposed phases.

800 It is conceivable that a threshold scheme is bootstrapped without prior agreement of who the  
801  $n$  parties/identifiers are (or even what is value of  $n$ ). However, said agreement problem may,  
802 in some system models, be a distributed-systems problem outside the scope of exploring the  
803 essential cryptographic thresholdization of the primitive at stake. Therefore, the assumption  
804 of initial agreement on  $n$  identifiers is a possibility, not a requirement. A submission that  
805 considers an additional preparatory phase for agreement of  $n$  and who the  $n$  parties are  
806 should try to present said phase modularly separated from the remaining threshold scheme.

807 **Beneficiaries.** For some operations, such as threshold keygen, the *beneficiaries* of the  
808 computation are the parties, who end with a new (secret sharing) state (possibly requiring  
809 agreement in the sense of “security with **unanimous** abort”), and/or an administrator (e.g.,  
810 who receives a new public key). For other operations, such as threshold signing, the  
811 beneficiary can be an external client who requested the computation, to obtain an output.

812 **Client interface.** The client may or may not be aware of (and be able to interact distinctively  
813 based on) the  $n$ -party threshold composition. This can be affected by the input/output (I/O)  
814 interface (see §2.3 of NIST-IR8214A). For example, a secret-sharing of the I/O can affect  
815 whether or not a client can separately send/receive input/output shares to/from each party.

816 **Intermediaries.** The possibility of **concurrent** execution requests must be considered. A  
817 baseline description can assume that there is a possibly malicious **proxy** that can: interme-  
818 diate the communication between clients and the threshold entity, and authorize requested  
819 operations (e.g., the signing of a message).

## 820 5.2.2. T2.2: Distributed Systems and Communication

821 As long as the interface and rules for composition are clear, the specification of a threshold  
822 scheme can (and is recommended to) decouple the description of (i) the building blocks  
823 (e.g., consensus, reliable broadcast) of classical distributed-systems, from (ii) the description  
824 of cryptographic operations needed to support the secure multiparty computation over (or  
825 of) a secret-shared key.

826 The specification of instantiations of building blocks that make use of weaker resources (e.g.,  
827 enabling broadcast based on point-to-point channels) can be provided by referencing existing  
828 specifications, while evaluating the impact of those replacements. Then, the provided open-  
829 source implementation (see Section 4.3) of the overall threshold scheme can include (with  
830 proper attribution) open-source code from the referenced existing implementation of the  
831 applicable building blocks. The protocol can also be described with various phases (e.g.,  
832 offline, online, secret resharing), which may have differentiated requirements.

833 A baseline description can make strong assumptions about the communication network,  
834 including synchrony and reliability of transmission. However, the proposal must discuss the  
835 pitfalls of deployment in environments with weaker guarantees (e.g., with asynchronous and  
836 unreliable channels), and possible mitigations.

837 Different threshold schemes may be better suited to different communication environments,  
838 with dependence on guarantees (or lack thereof) of **synchrony**, **broadcast**, and **reliability**. It  
839 is important to understand how security guarantees break across these environments.

## 840 5.2.3. T2.3: Adversary

841 The security analysis in S9 must consider a well-specified adversary, namely their goals and  
842 capabilities. In particular, the specification must consider an adversary that:

- 843 1. **[active]** is able to corrupt parties (up to one or various specified corruption thresholds),  
844 them controlling them to arbitrarily deviate from the prescribed multi-party protocol;
- 845 2. **[adaptive]** is able to decide which parties to corrupt after observing some of the  
846 protocol execution; and
- 847 3. **[mobile]** persistently continues (attempting to) corrupt parties across multiple execu-  
848 tions of the main protocol, possibly corrupting parties after they have been recovered  
849 from a previous corruption.

850 The concrete ways in which the adversary performs corruptions may be related to other  
851 system-model options (e.g., communication network). In practice, some of the adversary's

capabilities will be modeled as part of the idealization required in T3. The characterization of threshold security may vary across various ranges of acceptable corruption thresholds mentioned in item 1. Furthermore, the case of item 3 is intended to induce characterization of various levels of insecurity (e.g., which properties break and which ones do not) when acceptable thresholds are surpassed. The latter characterization may in particular be affected by the use of proactive recovery mechanisms (see Section T4.3).

### 5.3. T3: Security Idealization

As mentioned in Section 3.3, provable security is a fundamental component of how modern cryptography analyzes the security of proposed multi-party threshold schemes. Therefore, the present call includes a requirement to include a security idealization that supports a proof of security. Such idealization will encompass the security goals of the threshold scheme. That said, there are aspects of security analysis that overflow the scope of a proof/idealization and that should also be discussed.

A proposal of threshold scheme must be supported on a **simulation**-based and/or a **game**-based security formulation. This entails defining an ideal **functionality** (e.g., in the ideal-real simulation paradigm, within the universal composability framework) or/and an idealized adversarial **game** (or set of games). Since security analysis is a multi-dimensional exercise, it may include more than one form of idealization, and possibly even diverse proofs across different nuanced security properties or formulations.

A submission must include, in S9, a “security proof” that the proposed threshold scheme satisfies the proposed security formulation in a suitable adversarial context (see T4). Such proof can be given by showing “emulation” of the ideal functionality, or by showing that a non-negligible adversarial advantage in each security game implies breaking an assumption.

The security analysis must discuss which known useful properties are captured, and which ones are not, by the idealized security formulation. For example, even though availability is a desirable property, generically speaking, a security formulation with stronger emphasis on confidentiality and integrity may purposely specify that an adversary is allowed to abort protocol executions, so that the formulated security notion is achievable. As another example (now of an unsuitable formulation), a sole requirement of hiding and binding for a commitment scheme would not suffice for a use (e.g., committing bids in an auction) that would also require a non-malleability property.

In both cases (simulation and game-based), the security analysis should also discuss the security consequences of real implementation of idealized components. In particular, it must:



- 885     • identify the required cryptographic assumptions, and any possibly-idealized trusted  
886       components in the setup or operations;
- 887     • discuss the (in)security consequences of foreseen real instantiations of the setup and  
888       ideal components.

889 The “security analysis” (S9) asked in this call relates to the logical specification of the thresh-  
890 old scheme (S6–S8), and not to the submitted reference implementation (M2). Nonetheless,  
891 comments about implementation security are also welcome in the security analysis. Further  
892 details about implementation security can be included in S13.

#### 893 5.4. T4: Security Versus Adversaries

894 The security analysis in S9 must consider a well-specified adversary (see T2.3), namely their  
895 goals and capabilities. In consideration of the modeled adversary (see T2.3), a proposed  
896 threshold scheme must aim for certain security goals, particularly with regard to how the  
897 adversary corrupts up to a corruption threshold number  $f$  of parties.

##### 898 5.4.1. T4.1: Active Security (Against Active Corruptions)

899 Proposed threshold schemes **must** achieve **active security** (i.e., against active corruptions,  
900 which enable corrupted parties to “maliciously” deviate from the protocol), as opposed to  
901 *passive* only.

##### 902 5.4.2. T4.2: Adaptive Security (Against Adaptive Corruptions)

903 There is a strong preference for considering threshold schemes that achieve **adaptive**  
904 **security** (i.e., security against adaptively chosen corruptions), as opposed to *static* only,  
905 with respect to critical safety properties (e.g., unforgeability [NIST-IR8214B-ipd, §5.2.3] and  
906 key-secrecy). Therefore, submitted schemes should also aim for security against adaptive  
907 corruptions for the major safety properties of interest.

908 Adaptive security may pose significant challenges in formal proofs of security, depending  
909 on the security formulation. For example, while deniability of execution may in some  
910 cases be required for indistinguishability between ideal and real executions, the use of  
911 non-committing encryption to achieve it could be excessive without a necessary practical  
912 benefit. On the other extreme, a proposed protocol must not allow the major safety properties  
913 of interest to be trivially broken in case of adaptive corruptions, as in the classical example  
914 of a protocol that delegates all capabilities to a small quorum that is difficult to guess in  
915 advance, but whose overall corruption (by an adaptive adversary) would be disastrous.



916 The set of security formulations across submissions of threshold schemes (some possibly  
917 proving adaptive security based on unrealizable assumptions, such as a programmable  
918 random oracle) is expected to serve as reference material for public discussion. It is  
919 acceptable that certain security assurances (e.g., liveness and termination options) vary  
920 across different adversaries. For example, a security analysis may prove security against  
921 static corruptions with respect to some formulation (e.g., simulation-based), and then in  
922 complement show which fundamental security properties or attributes (e.g., unforgeability)  
923 remain preserved against adaptive corruptions in another formulation (e.g., game-based),  
924 even if some other security properties (e.g., some aspect of composability) are not preserved.

925 **Practical feasibility** is also needed. Feedback is welcome on security formulations and  
926 reference approaches that simultaneously enable both practical feasibility and security  
927 against adaptive corruptions, as well as possible acceptable tradeoffs.

#### 928 5.4.3. T4.3: Proactive Security (Against Mobile Attacks)

929 The proposed threshold schemes should be compatible with modular subprotocols / mech-  
930 anisms for **proactive** (and reactive) recovery, which attempt to recover possibly corrupted  
931 parties back to an uncorrupted state. This is especially important to better handle a persistent  
932 **mobile** adversary that continuously attempts to corrupt more parties. With respect to re-  
933 freshing secret shares, the solutions can be based on a modularized phase of secret-resharing  
934 (see T6), while also specifying the needed conditions (e.g., requirement of some initial/final  
935 agreement by a qualified quorum) for its integration.

#### 936 5.5. T5: Threshold Profiles

937 For each primitive (to be identified in S6, within the scope established in Sections 6 and 7)  
938 considered for thresholdization, it may be useful to consider differentiated solutions across  
939 possible threshold parametrizations. Therefore, it is useful to consider a “threshold profile”  
940 that defines, for certain threshold-related parameters, which parametrization ranges are  
941 suitable for secure operation. The threshold profile should characterize at least the total  
942 number ( $n$ ) of parties and the various thresholds ( $f$ ) of corruption and ( $k$ ) of participation.  
943 Table 3 proposes succinct labels for each default profile obtained from a restriction in the  
944 number of parties and the corruption threshold.

945 For convenience of discussion, the following nomenclature is defined to easily identify  
946 some default threshold profiles, based on the total number of parties and/or some corruption  
947 threshold ( $f$ ) assumed clear in the context.

- **Number  $n$  of parties:** (2) “two” for  $n = 2$ ; (3) “three” for  $n = 3$ ; (S) “small” for  $4 \leq n \leq 8$ ; (M) “medium” for  $9 \leq n \leq 64$ ; (L) “large” for  $65 \leq n \leq 1024$ ; and (E) “enormous” for  $n > 1024$ .
- **Corruption proportion  $f/n$ :** (D) “dishonest majority” for  $f \geq n/2$ ; (h) “honest majority” for  $f < n/2$ ; (H) “two-thirds honest majority”  $f < n/3$ .

**Table 3.** Labels for some template threshold profiles

Corruption proportion		Number of parties ( $n$ )					
$f/n$	Majority type	Two (2): $n = 2$	Three (3): $n = 3$	Small (S): $4 \leq n \leq 8$	Medium (M): $9 \leq n \leq 64$	Large (L): $65 \leq n \leq 1024$	Enormous (E): $n \geq 1025$
$\geq 1/2$	Dishonest (D)	$n2$	$n3fD$	$nSfD$	$nMfD$	$nLfD$	$nEfD$
$> 1/3$	Honest (h)	—	$n3fh$	$nSfh$	$nMfh$	$nLfh$	$nEfh$
$< 1/3$	2/3 Honest (H)	—	—	$nSfH$	$nMfH$	$nLfH$	$nEfH$

Note: the default profiles exclude the cases  $f = 0$  and  $f = n$ . Therefore: for the “two”-party profile (with  $n = 2$ ) — the usual **secure two-party computation (S2PC)** setting — only the “dishonest majority” case matters (with  $f = 1$ ); for the “three”-party profile, the 2/3 honest majority case does not apply. Other threshold profiles can be considered in concrete submissions. For example, some threshold schemes may have advantageous properties when considering an even stricter honest majority, such as more than 3/4 of honest parties.

A submission can focus on a single or on various threshold profiles. In particular, a protocol may be designed for *full threshold*, i.e., to ensure (for some range of number  $n$  of parties) some specific useful security notion regardless of the corruption threshold value  $f$  (with  $f < n$ ) that it is instantiated with. In some of such cases it may be especially relevant to distinguish between corruption threshold and participation-minus-1 threshold. For each submitted threshold scheme, the system model (S7) and the security analysis (S9) must:

- characterize its proposed threshold profile(s), including discussing the diversity of thresholds associated with various security properties; and
- characterize the breakdown that occurs when threshold-profile assumptions are broken.

**Note on alternatives access structures.** Depending on which secret-sharing schemes support the distributed computation, it is possible to consider monotone access structures (i.e., where the superset of a valid quorum is also a quorum) different from a simple threshold. The use of the traditional term “threshold” in this call is not meant to suppress possible submissions for other useful and properly-justified access structures.

979 **Motivating adoption.** There is value in identifying motivating applications for the adoption  
980 of threshold schemes in each threshold profile. Therefore, the submission should identify  
981 (in [S13](#)) use-cases for which the proposed threshold ranges are adequate.

## 982 5.6. T6: Building Blocks

983 A submission should identify and modularize the description of building blocks (gadgets)  
984 that can be securely replaced by other instantiations with similar interface. These may be  
985 useful across various threshold schemes across various submissions. While some future  
986 guidelines and recommendations documents may focus on gadgets, the decision to do so is  
987 likely to be subordinate to their utility for concrete threshold schemes.

988 **Example building blocks.** A notable building block is Shamir **secret sharing** (and Lagrange  
989 interpolation), either in the clear or homomorphically (e.g., “in the exponent”). Other secret  
990 sharing variants may also be useful, such as verifiable or publicly-verifiable secret-sharing.  
991 Other examples of gadgets include **garbled circuits**, **oblivious transfer**, **generation of**  
992 **correlated randomness**, **commitments**, **secret resharing** (possibly for new values  $f$  and  $n$ ),  
993 **multiplicative-to-additive share conversion**, **additively homomorphic encryption**, MPC  
994 or ZKP friendly hashing, some **zero-knowledge proofs**, **consensus** and **broadcast**.

995 **Modularized description.** To the extent possible, proposals of threshold schemes should  
996 modularize the description of gadgets. This means that a high-level description of the  
997 threshold scheme uses references to the interface and security properties of the gadgets, but  
998 not necessarily to low-level details. A lower level description can then be made for one (or  
999 more) possible instantiation of each needed gadget.

1000 **Modularized code.** The submitted open-source code (see Section [4.3](#)) must include code  
1001 for at least one instantiation of each used building block. If the proposed system model  
1002 depends on special hardware components (e.g., a router) beyond the threshold “parties”, the  
1003 submission should also include code for emulating the special component.

1004 The challenges faced in (i) implementing networking between parties can be significantly  
1005 different from those in (ii) implementing certain mathematical operations (cryptographic  
1006 building blocks) per party. Also, neglecting any of these can lead to serious vulnerabilities.  
1007 Therefore, it is strongly encouraged that there is a strong alignment between the proposed  
1008 system model (see [T2](#) in Section [5.2](#)) and the provided implementation (see Section [4.3](#)),  
1009 notwithstanding possible virtualizations to enable execution in a personal computer. For  
1010 example, if a system model relies on broadcast, then the provided implementation should  
1011 instantiate it in alignment with the assumptions of the proposed system model.

## 6. Cat1 primitives — Specified by NIST

Table 4 lists various Cat1 primitive-families of interest for thresholdization, organized in various “types” (subcategories): Signing (Section A.1); PKE (Section A.2); ECC-2KA (Section A.3); Symmetric (Section A.4); and Keygen (Section A.5). Within each type, each listed “primitive family” (itself identified with a more detailed subcategory index) may include several primitive variants (including ones not listed) and/or threshold modes, some of which are listed (non-exhaustively) in the third column of Table 4. A submission of threshold schemes fitting within a primitive family is not required to cover all indicated variants or modes, and may instead focus on a single one.

**Table 4.** Primitives of interest in subcategories of Cat1

Subcategory: Type	(Sub)subcategory #: Family of primitives	Some [Primitives] and/or {Threshold Modes}	Section in this call
C1.1: Signing	C1.1.1: EdDSA sign	[EdDSA, HashEdDSA] {Prob; Q-PR; F-PR (not FE); FE}	A.1.1
	C1.1.2: ECDSA sign	{Prob-FE; Q-PR; F-PR not-FE; PR-FE to Det-ECDSA}	A.1.2
	C1.1.3: RSADSA sign	[RSASSA-PSS; RSASSA-PKCS-v1.5]	A.1.3
C1.2: PKE	C1.2.1: RSA encryption	[RSASVE.Generate, RSA-OAEP.Encrypt] {SSI}	A.2.1
	C1.2.2: RSA decryption	[RSASVE.Recover, RSA-OAEP.Decrypt] {NSS, SSO}	A.2.2
C1.3: ECC-2KA	C1.3.1: ECC-CDH	{NSS, SSO}	A.3.1
	C1.3.2: ECC-MQV	[Full; One-pass] {NSS, SSO}	A.3.2
C1.4: Symmetric	C1.4.1: AES (en/de)cipher	[encipher, decipher]	A.4.1
	C1.4.2: KDM/KC (for 2KE)	[Hash, CMAC, HMAC, KMAC]	A.4.2
C1.5: Keygen	C1.5.1: ECC keygen	[For ECC-signing and ECC-2KA]	A.5.1
	C1.5.2: RSA keygen	[Just the modulus (mod); mod & keypair]	A.5.2
	C1.5.3: Bitstring keygen	[RBG for AES keygen, RSA-SVE, and nonces] {SSO}	A.5.3

Legend: 2KE = pair-wise key-establishment. Det = deterministic. FE = functionally equivalent. F-PR = fully PR (i.e., deterministic even if the quorum changes). KD/KC = key derivation and key confirmation mechanisms; NSS = input/output is not secret-shared (i.e., apart from the key); PKE = public-key encryption. PR = pseudorandom. Prob = probabilistic. RBG = random-bit generation. Q-PR = PR per quorum. SSI/SSO = secret-shared input/output (see §2.3 of NIST-IR8214A). SVE = secret-value encapsulation.

There are significant differences in threshold-friendliness and usefulness across the Cat1-primitives. For example, some symmetric-key primitives, such as HMAC and KMAC used for key-confirmation, are much less threshold-friendly than primitives based on public-key cryptography for signing and encryption/decryption. These differences are expected to affect the interest of stakeholders in submitting corresponding threshold schemes. Threshold-friendlier primitives can be considered in Cat2, as already conveyed in Table 2 in Section 3.2.

## 1045 **6.1. Input/Output (I/O) Interfaces**

1046 As discussed in §2.3 of [NIST-IR8214A](#), threshold schemes can be considered in various  
1047 modes with regard to the I/O interface. By default, a threshold keygen scheme produces a  
1048 secret-shared output (SSO), i.e., a secret-shared secret/private key, and (when applicable) a  
1049 corresponding not-secret-shared (NSS) public-key counterpart. Then, a subsequent threshold  
1050 operation (e.g., signing) uses the private/secret key in a secret-shared input (SSI) manner.  
1051 The mentioned secret-sharings (SSO and SSI) of the private/secret key are often left implicit.  
1052 However, the secret-sharing of other input/output (that may itself be subject to confidentiality  
1053 requirements) is relevant in some use cases, to hide said input/output from the threshold  
1054 entity. Some of these SSI/SSO modes are explicit in Table 4. For example:

- 1055 • a threshold decryption scheme can be in SSO mode to hide the decrypted plaintext;
- 1056 • a threshold public-key encryption (exceptional case where there is no private key) can  
1057 be in SSI mode to hide some secret key being encapsulated;
- 1058 • a threshold CDH or MQV ECC key-agreement primitive may produce a SSO to hide  
1059 the agreed key before it is subject to a final key-derivation (KD) transformation;
- 1060 • a threshold signature scheme can be in SSI mode to hide the message being signed  
1061 (not shown in Table 4).

1062 A submitted specification of a threshold scheme must unequivocally identify which I/O  
1063 parameters need to be in secret-shared form and which ones need not.

## 1064 **6.2. Cryptographic Parameters**

1065 Submitted threshold schemes should be implemented and evaluated with one set of pa-  
1066 rameters for security strength  $\kappa \approx 128$ , and another one for some security strength  $\kappa \in \approx$   
1067  $[224, 256]$ . Table 5 lists recommended options for cryptographic parameters.

### 1086 **6.2.1. Elliptic Curves, for ECC-related Primitives**

1087 NIST-approved curves for elliptic-curve cryptography are specified in [SP800-186-Draft](#).  
1088 There are various representations and curves over prime fields, including

- 1089 • Weierstrass: P-256, P-384, P-521, W-25519, W-448
- 1090 • Montgomery: Curve25519, Curve448
- 1091 • Twisted Edwards: Edwards25519, Edwards448, E448

**Table 5.** Recommended implementation parameters for Cat1 primitives

1068

1069	Parameter type	Primitives using said parameters	For $\kappa \approx 128$	For $\kappa \gtrsim 224$
1070	Elliptic curve	EdDSA signing and keygen	Edwards25519	Edwards448
1071		ECDSA signing and keygen	P-256	P-521
1072		ECC CDH/MQVfor 2KA, and keygen	{Curve25519, P-256}	{Curve448, P-521}
1073	RSA modulus size	RSADSA, RSA PKE, and their keygen	$ N  = 3,072$	$ N  \geq 11,264$ *
1074	RSA enc./ver. key	RSA-related	$2^{16} < e < 2^{256}$	$2^{16} < e < 2^{256}$
1075	Hash function	EdDSA signing	SHA-512	SHAKE256 (len 512, 912)
1076		ECDSA/RSADSA; HMAC for KDM/KC	SHA-256, SHA3-256,	SHA-512, SHA3-512
1077			SHA-512/256	
1078			SHAKE128 (len 256)	SHAKE256 (len 512)
1079	KMAC	for KDM and KC	KMAC128	KMAC256
1080	Cipher	KC (for RSA or ECC), encipher/decipher	AES-128	AES-256
1081	AES key-size	AES encipher/decipher/keygen/CMAC	$ k  = 128$	$ k  = 256$

1082 Legend:  $\kappa$  = standardized “security strength” (in bits). enc./ver. = encryption/verification. len = length.

1083 \* The RSA modulus length  $|N|$  must be a multiple of 8; this call further suggests that it be a multiple of 512.  
 1084 Approved hash functions or XOFs are specified in [FIPS-180-4](#), [FIPS-202](#), and [SP800-185](#), but only a subset  
 1085 of them are suggested in this call. A XOF with predetermined length (len) can also be called a hash function.

1092 A submission of threshold scheme for an ECC-based primitive should include an implemen-  
 1093 tation based on at least one curve for security level for  $\kappa \approx 128$ , and another for  $\kappa \gtrsim 224$ ,  
 1094 from the subsets detailed in Table 5. The curves W- $x$  (for some  $x$ ) and E448 do not appear  
 1095 in Table 5, as they are only intended for possible intermediate representations.

1096 Note that [SP800-186-Draft](#) also specifies curves over binary fields (in short-Weierstrass form,  
 1097 namely Koblitz curves (K-163, K-233, K-283, K-409, K-571) and some pseudorandom  
 1098 curves (B-163, B-233, B-283, B-409, B-571). However, these are for legacy-only appli-  
 1099 cations, and have been deprecated due to their limited adoption. Therefore, these are not  
 1100 recommended for submissions of threshold schemes.

1101 **Additive notation.** In elliptic-curve cryptography, it is customary to use additive group  
 1102 notation. There, a public key  $Q$  can be determined by a repeated sum of the base-point  $G$ ,  
 1103 a secret number  $d$  of times. The repeated-sum operation is (in additive notation) usually  
 1104 expressed as a multiplication by an integer. Thus, the private key  $d$  is the integer (not an  
 1105 elliptic curve element) needed to be multiplied with  $G$  to obtain  $Q = d \cdot G$ .

1106 **On the set of suggested curves for 2KA.** [SP800-56A-Rev3](#) (from 2018) considers (in  
 1107 its Table 24 in Appendix D) various curves for ECC key-agreement. Apart from Koblitz

(K- $x$ ) and pseudorandom (B- $x$ ) curves that have been deprecated by [SP800-186-Draft](#), the Weierstrass curves (P- $x$ ) remain valid. From the latter, P-256 and P-521 cover the cases for security levels  $\kappa \approx 128$  and  $\kappa \gtrsim 224$ . The recent [SP800-186-Draft](#) also specifies new Montgomery curves Curve25519 and Curve448, and references the IRTF [RFC7748](#) where those curves are suggested for use in 2KA. Despite their current potential for adoption, the older [SP800-56A-Rev3](#) does not include the new Montgomery curves (from the more recent [SP800-186-Draft](#)) in the list of approved curves for 2KA. Therefore, for Cat1-submissions of threshold schemes for ECC-2KA (subcategory [C1.3](#)): (i) the reference implementation should use at least the approved Weierstrass curves (P-256, P-521); (ii) a complementary suggestion is that Montgomery curves (Curve25519, Curve448) also be implemented to allow for a comparison across the uses of the two types of curves.

## 6.2.2. RSA Modulus, for RSA-related Primitives

A submission of threshold schemes for RSA-related primitives (for signing, key-encapsulation or decryption): should provide implementations with moduli of size  $|N| = 3072$  for  $\kappa \approx 128$ , and  $|N| \geq 11,264$  (or greater) for  $\kappa \approx 224$  (or greater, respectively). Note: [SP800-56B-Rev2](#) uses the symbol  $s$ , instead of  $\kappa$ , to denote the “security strength” (in bits).

The recommended RSA-modulus length  $|N|$  for security parameter  $\kappa \gtrsim 224$  was obtained, from exponential interpolation between the cases (specified in [SP800-57-P1-R5](#)) using  $|N_1| = 7680$  for  $\kappa_1 = 192$ , and  $N_2=15,360$  for  $\kappa_2 = 256$ , and rounding up to the nearest multiple of 512. The used formula is  $|N| = 512 \cdot \lceil |N_1| \cdot (\kappa/\kappa_1)^a / 512 \rceil$ , where  $a = \log_{(\kappa_2/\kappa_1)}(N_2/N_1)$ . This is also the value that would be obtained by rounding up the result provided by the FIPS 140-2 implementation guidance [[IG-FIPS-140-2](#), §7.5, page 125].

NIST-specified requirements for the prime factors of an RSA modulus, and their primality testing, are described in Appendices A.1 and C of [FIPS-186-5-Draft](#), for single-party generation. For threshold schemes that warrant different methods (e.g., direct biprimality testing), a rationale must be presented to convey why the used test (including the number of rounds) is appropriate. In particular, it is acceptable that the RSA modulus be biased toward being a Blum integer, i.e., with both primes being 3 mod 4.



## 1136 7. Cat2 Primitives — Not Specified by NIST

1137 Cat2 allows for submissions of threshold schemes for primitives that are not specified by  
1138 NIST. This category is aimed to allow for the consideration of primitives that are threshold-  
1139 friendlier than those in Cat1, and/or that have distinctive features, such as being based on  
1140 distinct cryptographic assumptions (possibly being quantum-resistant), or having advanced  
1141 functional features. Section 3.2 already enumerated the subcategories and listed some  
1142 examples (see Table 2). A submission in Cat2 must provide a thorough description of the  
1143 corresponding conventional (non-threshold) scheme that the primitive (being thresholdized)  
1144 is part of. For example: a submission of threshold scheme for a signing primitive not  
1145 specified by NIST must include a description of not only the conventional signing primitive  
1146 but also its corresponding verification and keygen primitives.

### 1147 7.1. “Regular” Primitives (Subcategories C2.1–C2.5)

1148 As already enumerated in Section 3.2 (including listed in Table 2), Cat2 covers five regular  
1149 types of primitives across subcategories C2.1 (for signing), C2.2 (for PKE), C2.3 (for  
1150 key-agreement), C2.4 (for symmetric-key and hashing primitives) and C2.5 (for keygen).

1151 Since selected candidates from the NIST PQC and Lightweight Cryptography (LWC) pro-  
1152 jects [Proj-PQC; Proj-LWC] are not yet standardized, possible threshold schemes for their  
1153 primitives can be presented in the scope of Cat2, specifically in their matching subcategories:  
1154 C2.1 (signatures) and C2.2 (public-key encryption) for PQC; C2.4 (symmetric-key and  
1155 hashing primitives) for LWC. However, the present call is also intended to elicit submissions  
1156 for threshold schemes for primitives that are threshold-friendlier. Submissions of threshold  
1157 schemes for quantum-resistant primitives should include a comparison with the security  
1158 levels (1–5) defined by the NIST PQC project [Proj-PQC].

1159 Subcategory C2.3, for single-party primitives for use in multi-party key-agreement, also  
1160 expects possible submissions of TF-QR type. Such submissions should demonstrate the  
1161 use of the thresholdized primitives in the scope of an actual key-agreement application.  
1162 Compared to NIST-standardized KA protocols, submissions in this sub-category may enable  
1163 improved KA schemes, justified based on different assumptions.

1164 **Note on PKE versus KA.** Primitives within subcategory C2.2 for PKE can be used  
1165 for multi-party key-establishment protocols, by allowing the confidential transmission  
1166 of a contribution to a key. The subcategory C2.3 for KA (within Cat2) is intended for  
1167 complementary primitives, such as those that may enable key-exchange protocols *a la*



Diffie-Hellman, though possibly based on different assumptions (e.g., to be QR) or for more than two parties. Therefore, the subcategory C2.3 for KA excludes the key-transport-only mechanisms (whose main cryptographic primitive is already scoped by PKE).

## 7.2. “Other” Primitives/Schemes (Subcategories C2.6–C2.8)

Beyond the “regular” type of primitives (covered by Cat1 and Cat2), there are “other” types of primitives covered by Cat2, namely “advanced” primitives (C2.6; see Sections 7.2.1 and A.6), “ZKPoKs” (C2.7; see Sections 7.2.2 and A.7) and “auxiliary gadgets” (C2.8; see Sections 7.2.3 and A.8). The subcategories for ZKPoK (C2.7) and gadgets (C2.8) are meant to allow for the submission of primitives that can support the threshold setting. Such a submission requires the specification of a conventional (non-threshold) primitive (see S6), but (in contrast with other subcategories) the specification of a threshold scheme is optional.

### 7.2.1. Cat2 subcategory C2.6: “Advanced”

Subcategory C2.6 (see more details in Section A.6) is suited for primitives with *advanced functional features* that are not covered by current NIST standards. For example, an encryption scheme may allow (i) homomorphically performing operations over encrypted data (possible with fully-homomorphic encryption), or (ii) selectively restricting the ability for decryption to designated sets of recipients (possible with identity-based and attribute-based encryption). A submission in subcategory C2.6 should present a strong rationale for the utility of the enhanced features, compared to what is possible with primitives in the other subcategories. Since quantum resistance is a strongly desirable feature, a submission without such a property is encouraged to specifically present rationale about the lack of good TF-QR alternatives.

### 7.2.2. Cat2 subcategory C2.7: ZKPoK

Subcategory C2.7 (see more details in Section A.7) allows for the submission of **zero-knowledge proofs of knowledge** (ZKPoKs) that can support the threshold environment. For example, they may be useful to prove knowledge of a secret/private key or input that is consistent with:

- a public-key and/or with the public commitments of secret-shares;
- the output of a cryptographic operation (e.g., public-key encryption, AES enciphering, or KDM hashing), when the input was secret-shared and committed.

1198 The generation of a ZKPoK can be considered both in conventional (non-threshold) and in  
1199 threshold forms. For example:

- 1200 • **[Conventional generation]** A dealer (single-party) of a secret-sharing (SS) can  
1201 produce a ZKPoK that enables the various parties of a threshold entity (recipients of  
1202 secret-shares) to non-interactively verify that the SS is adequate;
- 1203 • **[Threshold generation]** The set of parties that interacted in a DKG to obtain a secret-  
1204 sharing of a secret/private-key, and when applicable also obtain a corresponding  
1205 public-key, can interact in an MPC to distributively generate a ZKPoK string that  
1206 proves access to (i.e., knowledge of, albeit in a threshold manner and despite the secret-  
1207 sharing aspect possibly remaining hidden from the proof) an adequate secret/private  
1208 key consistent with a corresponding public commitment (possibly the public key) of  
1209 the given threshold scheme.

1210 (Note that the latter example is dissociated from a conceivable proof of distributed  
1211 generation of a key, which can be considered if tied to public keys of the intervening  
1212 parties, believed to not reveal their private keys.)

1213 The above two examples have similarities with, respectively, (i) verifiable secret sharing  
1214 (VSS), which can also be extended to publicly verifiable secret-sharing (PVSS), and (ii)  
1215 publicly verifiable MPC. Said verifiable features are welcome in submitted threshold schemes,  
1216 and may (preferably) be included as part of a submission more focused on one of the other  
1217 subcategories, while identifying the applicability of the ZKPoK to the present subcategory.  
1218 A submission that simply focuses in subcategory [C2.7](#) must specify at least a conventional  
1219 ZKPoK, and may (optionally) specify a corresponding threshold version thereof.

### 1220 7.2.3. Cat2 subcategory [C2.8](#): Auxiliary Gadgets

1221 Subcategory [C2.8](#) (see more details in Section [A.8](#)) allows for the submission of specifi-  
1222 cations of other auxiliary primitives, here called *gadgets*. They may be auxiliary in their  
1223 conventional (non-threshold) form and/or in a threshold form. Gadgets can be modularized  
1224 in the submission of a higher-level threshold scheme associated with another subcategory  
1225 within Cat1 or [C2.1–C2.7](#). Such modularization is already recommended by criterion [T6](#)  
1226 (in Section [5.6](#)) for various gadgets (e.g., those enumerated in §4.5.2 of [NIST-IR8214B-ipd](#)  
1227 and §5.3.1 of [NIST-IR8214A](#)) whose underlying primitives (e.g., garbled-circuit generation,  
1228 garbled circuit evaluation, commit, decommit) are not themselves thresholdized.

## 1229 A. Details for Subcategories and Primitives of Interest

### 1230 A.1. Subcategory C1.1: Cat1 Signing

1231 The three [Cat1](#)-signing primitives of interest are from EdDSA, ECDSA, and RSADSA.  
1232 Submissions in this subcategory should take in consideration the aspects of unforgeability  
1233 and threshold security mentioned in [NIST-IR8214B-ipd](#) (while some aspects are specific to  
1234 EdDSA, others are applicable to generic signature schemes). For example, it is useful to  
1235 differentiate between regular unforgeability and strong unforgeability.

#### 1236 A.1.1. Subcategory C1.1.1: EdDSA Signing

1237 EdDSA is specified in §7 of [FIPS-186-5-Draft](#). The default signing mode is pseudorandom,  
1238 determining the secret nonce  $r$  as a hash output whose pre-image includes a nonce-derivation  
1239 key  $v$ . Ignoring some encoding details, the algorithm for EdDSA signing  $\text{Sign}_n[s, v](M)$   
1240 of a message  $M$  outputs a signature  $\sigma = (R, S)$ , where  $R = r \cdot G$ ,  $G$  is the conventioned  
1241 base-point of the elliptic curve,  $r = H(v, M)$ ,  $H$  represents a cryptographic hash function,  
1242  $S = r + \chi \cdot s$ ,  $\chi = H(R, Q, M)$  is the “challenge”, and  $s$  is the private signing key (integer)  
1243 needed to be multiplied with  $G$  to obtain the public-key  $Q$ .

1244 A submission of threshold scheme for EdDSA signing: can choose to implement just one  
1245 of or both HashEdDSA and EdDSA types (defining whether or not the message is “pre-  
1246 hashed”); should provide implementations with curves Edwards25519 (for  $\kappa \approx 128$ ) and  
1247 Edwards448 (for  $\kappa \approx 224$ ), which are specified in [SP800-186-Draft](#); and must include only  
1248 schemes that are interchangeable with regard to EdDSA verification (see related notes in  
1249 [NIST-IR8214B-ipd](#)). With respect to nonce generation, submissions are expected to include  
1250 one or more of the following modes:

- 1251 1. **Probabilistic** (via a random or hybrid contribution per party)
- 1252 2. **Pseudo-random per quorum** (via a ZKP of pseudorandom contribution per party)
- 1253 3. **Pseudo-random** (based on a threshold-friendly PRF)
- 1254 4. **Functionally equivalent to HashEdDSA** (via MPC hashing)

1255 **Note.** An SSI mode for threshold signing is costly because it requires a distributed com-  
1256 putation of a threshold-non-friendly hash of the message. However, if the regular NSS  
1257 mode already requires such type of difficult computation (which is the case in functionally-  
1258 equivalent EdDSA threshold signing), then the SSI mode may be achieved with a simple  
1259 extension, using the gadgets already required for the NSS mode.

### 1260 A.1.2. Subcategory C1.1.2: ECDSA Signing

1261 ECDSA is specified in §6 of [FIPS-186-5-Draft](#). The default signing mode is probabilistic  
1262 (§6.3.1), but there is also a deterministic ECDSA mode (§6.3.2). Table 6 shows how the  
1263 meanings of some symbols change significantly between EdDSA and ECDSA.

1264 **Table 6.** Notation of EdDSA versus ECDSA (in Draft FIPS 186-5)

1265	Element’s role	In EdDSA	In ECDSA
1266	Signature	$(R, S)$	$(r, s)$
1267	Private <sup>†</sup> key	$s$	$d$
1268	Secret nonce	$r$	$k$
1269	[Final] <sup>‡</sup> nonce commitment	$R$	$r$
1270	Challenge	$\chi$	$e$

1271 <sup>†</sup> EdDSA also uses  $d$ , but for the precursor private-key from which the signing key  $s$  and another  
1272 nonce-derivation key are obtained. <sup>‡</sup> The use of [final] is to convey that it is the actual value output in the  
1273 signature. It is an encoding of other intermediate computed values that are themselves also commitments  
1274 to the nonce. In particular, in ECDSA one of the intermediate values is denoted with symbol  $R$ .

1275 Ignoring some encoding details, the algorithm for ECDSA signing  $\text{Sign}_n[d](M)$  of a mes-  
1276 sage  $M$  outputs a signature  $\sigma = (r, s)$ , where  $d$  is the private signing key (the integer  
1277 needed to be multiplied with the base-point  $G$  to obtain the public-key  $Q$ ); the “challenge”  
1278  $e = \text{Encode}_n^{(1)}(\text{Hash}(M))$  is an encoding (mod  $n$ ) of the hash of the message being signed;  
1279  $k \leftarrow^{\$} [1, \dots, n-1]$  is (in the probabilistic version) a uniformly selected nonce that needs to  
1280 remain secret;  $R = k \cdot G$  is the “nonce commitment” and  $r = \text{Encode}_n^{(2)}(R)$  is a corresponding  
1281 encoding (mod  $n$ ); and  $s = k^{-1} \cdot (e + r \cdot d) \pmod{n}$ .

1282 A submitted threshold scheme for ECDSA signing should provide an implementation  
1283 with at least one parametrization for  $\kappa \approx 128$  and another for  $\kappa \gtrsim 224$ , with parameters  
1284 recommended in Table 5. With respect to nonce generation, submissions are expected to  
1285 include at least one of the following modes:

- 1286 1. **Probabilistic** (via random or hybrid contributions per party)
- 1287 2. **Pseudo-random per quorum** (via a ZKP of pseudorandom contribution per party)
- 1288 3. **Pseudo-random** (based on a threshold-friendly PRF)
- 1289 4. **Pseudo-random functionally equivalent to Deterministic ECDSA** (via MPC hashing)

1290 **Note on SSI-signing:** In the case of SSI-signing for Deterministic ECDSA, the client  
1291 can directly provide a secret-shared challenge (the hash  $e$  of the message), whereas in  
1292 (Deterministic) EdDSA the pseudorandom challenge  $\chi$  requires knowledge of a nonce

1293 commitment that depends on a private element not known by the client. Note that signature  
1294 verification still requires the ability to hash the message.

### 1295 **A.1.3. Subcategory C1.1.3: RSADSA Signing**

1296 RSA signature modes are specified in §5.4 of [FIPS-186-5-Draft](#), by reference to IETF [RFC8017](#).  
1297 A submission for the RSADSA signing family is expected to implement a threshold signature  
1298 scheme that is interchangeable with at least one of the following modes:

- 1299 1. RSASSA-PSS (probabilistic signature scheme), using an approved hash function or XOF
- 1300 2. RSASSA-PKCS-v1.5 (deterministic), using an approved hash function

### 1301 **A.1.4. Signing in Secret-Shared-Input (SSI) Mode**

1302 In an SSI-signing mode, no single-party (nor any collusion up to a certain number of parties)  
1303 of the threshold entity will learn the hash of the message. This is akin, though not the same  
1304 as, what is achieved with blind signatures. The difference is that in the threshold setting it is  
1305 possible that a large enough collusion of parties is able to reconstruct the input message.

1306 The SSI mode may be of use, for example, for private-preserving time-stamping, producing  
1307 a certificate interchangeable with those produced by the conventional protocol where the  
1308 authority learns the hash of the document being timestamped.

1309 The threshold-generation of signatures in SSI mode may pose challenges with regard to  
1310 unforgeability. For example, a protocol must prevent that a malicious party that maliciously  
1311 changes their secret-share would affect the overall message being signed, i.e., must prevent  
1312 the signing of a message whose signature has not been requested. Such challenges may  
1313 be resolved based on various techniques, including zero-knowledge proofs, or based on  
1314 verifiability or error correction properties of the secret-sharing. For example, each party can  
1315 prove that their interaction in the distributed computation is consistent with a secret-share  
1316 that has been certified by the client, with regard to the ongoing signing session.

### 1317 **A.2. Subcategory C1.2: Cat1 Public-Key Encryption (PKE)**

1318 The PKE cryptosystem of interest is RSA. The main use case considered for RSA encryption/  
1319 decryption is pair-wise key-establishment (2KE), as specified in [SP800-56B-Rev2](#). 2KE  
1320 can take the form of a key-agreement (KA) type of protocol (with contributions from both  
1321 parties) or be more simply based on key-transport (KT) type of protocol (with contribution  
1322 from a single party). For RSA-based instantiations, both types of protocol rely on secret-  
1323 value encapsulation (SVE), where RSA encryption is used to encapsulate a secret value

1324  $k$  (also denoted as a plaintext  $m$ ) into a ciphertext  $c$ , which is then sent to another party  
1325 for decryption. Ignoring some encoding details, the low-level RSA-based cryptographic  
1326 primitives of interest are:

- 1327 • **RSA encryption primitive (RSAEP):** Encryption  $c = m^e \bmod N$  (transforming a  
1328 plaintext  $m$  into a ciphertext  $c$ ). A threshold version of it uses a secret-shared input  $m$   
1329 (SSI) and a not-secret-shared public encryption key.
- 1330 • **RSA decryption primitive (RSADP):** Decryption  $m = c^d \bmod N$ . A threshold version  
1331 of it uses a secret-shared private-key  $d$  (which is never reconstructed); the threshold  
1332 operation produces an output that is either secret-shared (SSO) or not (NSS).

1333 Additional relevant primitives include:

- 1334 • Generation of an RSA modulus and/or key-pair (see Section A.5.2).
- 1335 • Generation of a random bit-string (see Section A.5.3).

1336 The values generated in SSO mode are for subsequent consumption in SSI mode.

#### 1337 **A.2.1. Subcategory C1.2.1: RSA Encryption (of a Secret-Value)**

1338 Threshold schemes in this call are intended to operate over secret-shared material. Therefore,  
1339 in the case of public-key encryption the secret-sharing does not usually apply to the public  
1340 key. However, the application of key-encapsulation for key-transport/agreement uses the  
1341 plaintext itself (being encrypted) as a value whose confidentiality requirement may warrant  
1342 threshold protection. By default, a threshold scheme for such encryption will be in “secret-  
1343 shared input” (SSI) mode (see [NIST-IR8214A]) with regard to the value being encrypted,  
1344 but will not secret-share the public key (to be known by every party).

1345 The basic **RSA encryption primitive (RSAEP)** computes a ciphertext  $c = m^e \bmod N$ ,  
1346 where  $m$  is a secret plaintext,  $e$  is the public encryption key, and  $N$  is the public modulus.  
1347 The goal is to compute  $c$  from a secret sharing  $[m]$  of  $m$ . For interchangeability with regard to  
1348 a subsequent decryption, an actual full-fledged threshold scheme for RSA key encapsulation  
1349 should consider all of the appropriate encoding and padding details. In SP800-56B-Rev2, the  
1350 primitive RSAEP (§7.1.1) is specified for use within two higher-level primitives:

- 1351 1. RSASVE.Generate (§7.2.1.2): RSA for **S**ecret-**V**alue **E**ncapsulation (which also  
1352 includes the generation of the random key to encapsulate)
- 1353 2. RSA-OAEP.Encrypt (§7.2.2.3): RSA with **O**ptimal **A**symmetric **E**ncryption **P**adding

### 1354 **A.2.2. Subcategory C1.2.2: RSA Decryption**

1355 [SP800-56B-Rev2](#) specifies the use of RSA decryption in two higher-level primitives:

- 1356 1. RSASVE.Recover (§7.2.1.3): Secret-Value Encapsulation recovery
- 1357 2. RSA-OAEP.Decrypt (§7.2.2.4): Optimal Asymmetric Encryption Padding decryption

1358 The RSA decryption primitive, RSADP([privKey](#),  $c$ ), used to decrypt a ciphertext  $c$ , accepts  
1359 the private decryption key [privKey](#) [[SP800-56B-Rev2](#), §6.2.2] in three possible formats:

- 1360 1. Basic format:  $(n, d)$
- 1361 2. Prime-factor format:  $(p, q, d)$
- 1362 3. Chinese-remainder theorem (CRT) format:  $(n, e, d, p, q, dP, dQ, qInv)$

1363 The notation [[SP800-56B-Rev2](#), §3.2] is as follows:  $n$  is the public modulus;  $(p, q)$  is the pair  
1364 of secret prime factors of  $n$ ;  $d$  is the private decryption key;  $e$  is the public encryption key;  
1365  $dP$  is  $d \bmod (p - 1)$ ;  $dQ$  is  $d \bmod (q - 1)$ ; and  $qInv$  is the inverse of  $q \bmod p$ .

### 1366 **A.2.3. Implementation Recommendations and Options**

1367 A submitted threshold scheme for RSA encryption or decryption primitives should include  
1368 an implementation in the scope of an RSA-based 2KE protocol, as follows:

- 1369 • With an instantiation for  $\kappa \approx 128$  and another for  $\kappa \gtrsim 224$  (see Table 5).
- 1370 • Showcasing at least one of the key-establishment protocols listed in Table 7, with at  
1371 least one of the parties ( $U$ , or  $V$ ) being threshold-decentralized;
- 1372 • If implementing threshold RSADP:
  - 1373 – secret-sharing the decryption key, for at least one of the three approved formats  
1374 (Section A.2.2); the public elements ( $n$  and  $e$ ) do not need to be secret shared;
  - 1375 – outputting the plaintext (the key that was encapsulated) in one of two forms:  
1376 secret-shared, or not secret-shared.
- 1377 • If implementing threshold RSAEP: using an SSI mode for the plaintext.

1378 **The various RSA-2KE schemes.** [SP800-56B-Rev2](#) specifies various RSA-2KE schemes.  
1379 Two are of the *key agreement* (KA) type (obtaining contributions from both parties), whereas  
1380 another one is based on *key transport* (KT) using a contribution from a single party. Table 7  
1381 lists, across these three schemes, the corresponding RSA-based operations (excluding  
1382 needed RSA key-pair generation). Each of the listed schemes allows for a basic version,



and a version with key confirmation (unilateral or bilateral, not based on RSA). The KDM operation specified for KA schemes is not RSA based.

**Table 7.** RSA-based primitives per party per RSA-2KE scheme

Type	Scheme	§ in SP 800-56B-Rev2	Party	RSA-based primitive	KDM needed?
KA	KTS1	§8.2	1st contributor ( $U$ )	RSASVE.Generate	Yes
			2nd contributor ( $V$ )	RSASVE.Recover	
	KTS2	§8.3	Any	RSASVE.{Generate & Recover}	
KT	KTS-OAEP	§9.2	Sender ( $U$ )	RSA-OAEP.Encrypt	No
			Receiver ( $V$ )	RSA-OAEP.Decrypt	

In KTS1, one party ( $U$ ) uses RSASVE.Generate to generate and encrypt a secret value  $Z$ , and the other party ( $V$ ) uses RSASVE.Recover to decrypt  $Z$ . The latter party then contributes a non-encrypted nonce  $N_V$ . (Per §5.4 of SP800-56B-Rev2, the nonce used in KTS1 should be random.) Both the secret value and the nonce are then used as input to a KDM, which produces a final agreed key  $k$  (not to be confused with the nonce  $k$  of ECDSA). In KTS2, the clear-text nonce from party  $V$  is replaced with an encapsulated key, therefore requiring both parties to implement both RSASVE.Generate and RSASVE.Recover. Both KTS1 and KTS2 include a subsequent KDM, either in a one-step version or a two-step version, which transforms the pair of contributions ( $Z$  and  $N_V$ ) into a final derived key  $k$ . A threshold keygen can consider the generation of  $Z$  and/or  $N_V$  in SSO mode Section A.5.3, if they are to then be consumed in SSI mode by the subsequent KDM.

The KTS-OAEP scheme does not use a KDM. Instead, the output key is decided by one of the parties, who then sends it encrypted to the other party. The threshold modes of interest for KTS-OAEP depend on the primitive, as follows:

- RSA-OAEP.Encrypt with the plaintext (a key to be encapsulated) in SSI mode.
- RSA-OAEP.Decrypt with the plaintext (the key that was encapsulated) in SSO mode.

Each 2KE scheme can be implemented in either a basic form (without key confirmation), or with KC in either a unilateral or bilateral manner. Both KDM and KC primitives rely on hash-functions of symmetric-key cryptography (see Section A.4.2).

SP800-56B-Rev2 also specifies that any of the mentioned RSA-2KE schemes (KTS1, KTS2, and KTS-OAEP) can be followed by a key transport where the established key is wrapped



with an approved (symmetric-key based) key-wrapping algorithm [SP800-38F]. However, threshold-wise said key-wrapping algorithms are more-unfriendly than KTS-OAEP.

**On the ability to bias the key in a 2KE protocol.** The various mentioned NIST-specified protocols allow one of the parties to significantly bias the result. Specifically, the second contributor party in the KTS1 and KTS2 protocols can brute-force its contribution to bias several bits (e.g., 40 bits, at a parallelizable computational cost of approximately  $2^{40}$  KDM operations). In KTS-OAEP the sender fully determines the key being transported. This is in contrast with Blum-style coin-flipping protocols, where the contribution from each party is only revealed once the contribution from the other party is committed to, thus implying that an honest party can guarantee that the output is not biased (up to abort by the other party).

### A.3. Subcategory C1.3: Cat1 ECC Primitives for Pair-Wise Key-Agreement (2KA)

**Pair-wise key-agreement (2KA).** SP800-56A-Rev3 specifies various pair-wise (i.e., two-party) key-establishment (2KE) schemes of the KA-type (where the final key depends on contributions from the two parties), based on discrete logarithm cryptography. In a 2KA scheme, each party uses their own private key(s) and the public key(s) from the other party, to first obtain an intermediate common secret  $Z$ , and then applies a transformation to obtain a final key (called *DerivedKeyingMaterial*)  $k$  that is equal to the one obtained by the other party (not to be confused with the nonce  $k$  of ECDSA).

In some NIST publications the intermediate secret  $Z$  is referred to as a “shared” secret, meaning it is known by both parties of the 2KA. This should not be confused with the case of a “secret-shared”  $Z$  when “thresholdizing” (i.e., decentralizing) one of the original parties.

Each 2KA protocol specified in SP800-56A-Rev3 can be described with up to three phases:

1. **A public-key cryptography (PKC) phase**, where the parties interact to determine an intermediate common secret  $Z$ .
2. **An asymmetric-key cryptography phase**, where each individual party uses a *key-derivation mechanism* (KDM) to derive a final key  $k$ .
3. **An optional key confirmation (KC) phase**, based on comparison of message authentication code (MAC) tags, which allows at least one of the parties to confirm that their obtained key is equal to the key of the other party.

The subcategory C1.3 (2KA) of Cat1 in this call is only focused on the PKC primitives used in the initial phase, namely the Cofactor Diffie-Hellman (CDH) or Menezes-Qu-Vanstone (MQV) primitives. However, a submission of a threshold scheme for such a primitive should be demonstrated in an implementation of a full-fledged 2KA protocol. Therefore, this section

also provides some context about the KDM and (the optional) KC operations, whose possible thresholdization is considered in Section A.4.2.

**ECC scope.** From the schemes in SP800-56A-Rev3, Cat1 only includes those based on ECC, which are implementable with elliptic curves specified in SP800-186-Draft. Table 5 in Section 6.2 lists the curves of interest. 2KA based on finite field cryptography (FFC) is left out of scope, following the trend of deprecating FFC in favor of more succinct ECC, as done in FIPS-186-5-Draft (which deprecated DSA in favor of ECDSA). The seven 2KA schemes in scope are listed in Table 8 and can be classified based on three factors:

- the underlying ECC primitive: CDH or MQV.
- the number of ephemeral (e) keys (2, 1 or 0),
- the number of static (s) keys (2, 1 or 0); and

**Table 8.** Seven ECC-2KA schemes

Primitive ( $f$ )	$e$	$s$	Scheme	Intermediate secret $Z$ ("agreed" by $U$ and $V$ )	§ in SP 800 -56A-Rev3
ECC CDH	2	2	(Cofactor) Full Unified Model	$f(e_U, E_V)    f(s_U, S_V)$	§6.1.1.2
	2	0	(Cofactor) Ephemeral Unified model	$f(e_U, E_V)$	§6.1.2.2
	1	2	(Cofactor) One-Pass Unified Model	$f(e_U, E_V)    f(e_U, S_V)$	§6.2.1.2
	1	1	(Cofactor) One-Pass Diffie-Hellman	$f(e_U, S_V)$	§6.2.2.2
	0	2	(Cofactor) Static Unified Model	$f(s_U, S_V)$	§6.3.2
ECC MQV	2	2	Full MQV	$f(s_U, S_V, e_U, E_U, E_V)$	§6.1.1.4
	1	2	One-Pass MQV	$f(s_U, S_V, e_U, E_U, S_V)$	§6.2.1.4

**Legend:**  $||$  = concatenation.  $§$  = section in another document.  $e$  = number of generated *ephemeral* key pairs.  $f$  = symbol representing the ECC primitive (CDH or MQV).  $s$  = number of generated *static* key pairs;  $U$  and  $V$  = the two parties in the 2KA protocol. Let  $A$  represent one of the parties ( $U$  or  $V$ ). **Abbreviated notation for keys:**  $e_A$  ( $= d_{e,A}$ ) and  $E_A$  ( $= Q_{e,A}$ ) are the *ephemeral* private and public keys of party  $A$ ;  $s_A$  ( $= d_{s,A}$ ) and  $S_A$  ( $= Q_{s,A}$ ) are the *static* private and public keys of party  $A$ . The primitive  $f$  makes use of additional parameters not shown here.

**Interchangeability scope.** Regardless of the decentralization of any party, a 2KA scheme is already a protocol between two parties that intend to obtain a commonly agreed secret. Therefore, when considering a threshold scheme for a Cat1-primitive of a 2KA protocol, the interchangeability requirement is narrowed to “functional equivalence”. This ensures that the output secret (albeit possibly in secret-shared format) on one decentralized side will be equal to the one obtained by the other (possibly legacy) party in the 2KA interaction. Cat2

(see Section 7) allows for interchangeability in a broader sense, assuming that both parties interacting in the 2KA can agree on the new subsequent (KD/KC) mechanisms.

**Single-party primitives.** The objects of thresholdization are the primitives (see Table 9) computed by each individual party in the 2KA protocol. Each of these primitives has private/secret key-material in the input or/and output. The threshold protection provided to the keys handled by one side of the ECC-2KA depends on which primitives are thresholdized.

**Table 9.** ECC-2KA primitives of interest for thresholdization

Primitive	Secret input?	Secret output?	Threshold friendly?	Section in SP800-56A-Rev3	Section in this call
ECC keygen: get key-pair ( $d, Q$ )	—	Yes	Yes	§5.6.1.2	A.5.1
ECC CDH/MQV: $Z = f(d_A, Q_B, \dots)$	Yes	Yes	Yes	§5.7	A.3.1/2
Key derivation: $k = \text{KDM}(Z, \dots)$	Yes	Yes	No	§5.8	A.4.2
Key confirmation: $\text{KC}(Z, \dots)$	Yes	—	No	§5.9	A.4.2

Legend:  $d$  = private key.  $f$  = CDH or MQV transformation (primitive).  $k$  = final secret established by both parties. KC = “key confirmation” pseudorandom function, to allow comparison between  $A$  and  $B$ . KDM = “key derivation mechanism” function.  $Q$  = public key.  $Z$  = intermediate secret (before KDM) computed by both parties.

A threshold scheme for an ECC CDH/MQV primitive allows for confidentiality of the private key  $d$ . This can be useful even if the intermediate secret  $Z$  is reconstructed due to a subsequent non-thresholdized KDM. Conversely, in a full-fledged thresholdization of the sequence of 2KA primitives, the output  $Z$  of the ECC CDH/MQV primitive would be secret-shared (i.e., SSO mode), to serve as input to the subsequent threshold KDM phase.

The ECC-2KA “type” includes only the ECC primitives that produce the intermediate secret  $Z$ , from secret-shared ECC private keys (static or ephemeral). There are two such primitives: ECC-CDH (Section A.3.1) and ECC-MQV (Section A.3.2). The ECC key-gen and KDM/KC primitives are respectively considered in Sections A.5.1 and A.4.2.

**Submissions.** A submitted threshold scheme for an ECC CDH or MQV primitive should:

- Evaluate it for at least one curve for  $\kappa \approx 128$ , and another for  $\kappa \in \approx [224, 256]$  — see Table 5 in Section 6.2.
- Showcase the execution of at least one of the seven 2KA ECC-based schemes (see Table 8), with at least one decentralized party ( $A$ ,  $B$ , or both) using secret-shared private keys in the threshold ECC CDH/MQV computation. The implementation should also include the KDM (and optionally the) KC procedures, either threshold (see

1508 Section A.4.2, if the threshold ECC CDH/MQV is in SSO mode) or non-threshold. In  
1509 other words, the ECC CDH/MQV output may or not be secret-shared, depending on  
1510 whether or not the subsequent KDM/KC primitive is thresholdized.

### 1511 A.3.1. Subcategory C1.3.1: ECC-CDH Primitive

1512 With a decentralized party  $A$  (which can be  $U$  or  $V$ ), the ECC-CDH primitive is as follows:

- 1513 • **Secret-shared input:**
  - 1514 –  $[d_A]$  (secret sharing of private key of party  $A$ )
- 1515 • **Public input:** (known to every party of the decentralized entity representing  $A$ )
  - 1516 –  $Q_B$  (the public key of party  $B$ );
- 1517 • **Secret-shared output:** Secret sharing  $[Z]$  of a secret  $Z = \text{Encode}(P)$ , where:
  - 1518 –  $P = (h \cdot d_A) \cdot Q_B$  (where  $h$  is the cofactor)
  - 1519 –  $\text{Encode}$  is an encoding that does a field-element-to-byte string conversion of the
  - 1520  $x$ -coordinate of the input.

1521 The output is distributively computed in a way that  $Z$  remains threshold confidential.

### 1522 A.3.2. Subcategory C1.3.2: ECC-MQV Primitive

1523 With a decentralized party  $A$  (which can be  $U$  or  $V$ ), the ECC-MQV primitive is as follows:

- 1524 • **Secret-shared input:**
  - 1525 –  $[d_{s,A}], [d_{e,A}]$  (secret sharings of the static and ephemeral private keys of party  $A$ )
- 1526 • **Public input:** (known to every party of the decentralized entity representing  $A$ )
  - 1527 –  $Q_{e,A}$  (the ephemeral public key of party  $A$ );
  - 1528 –  $Q_{s,B}$  and  $Q_{e,B}$  (the static and ephemeral public keys of party  $B$ )
- 1529 • **Secret-shared output:** Secret sharing  $[Z]$  of a secret  $Z = \text{Encode}(P)$ , where:
  - 1530 –  $P = h \cdot \text{impsig}_A \cdot (\text{avf}(Q_{e,B}) \cdot Q_{s,b})$ ;
  - 1531 –  $\text{impsig}_A = (d_{e,a} + \text{avf}(Q_{e,A}) \cdot d_{s,A}) \bmod n$ ;
  - 1532 –  $\text{avf}(Q)$  is an integer associated to a public key  $Q$ , computed via an “Associate
  - 1533 Value Function” ([SP800-56A-Rev3, §5.7.2.2]);

1534           – *Encode* is the same encoding as defined for ECC CDH.

1535 There are two possible implementation forms for the ECC MQV primitive:

- 1536       1. The **full form** ([SP800-56A-Rev3, §5.7.2.3.1]), implemented as described above, where  
1537       both static and ephemeral keys exist and are distinct.
- 1538       2. The **one-pass form** ([SP800-56A-Rev3, §5.7.2.3.2]), where exactly one other party (*A*  
1539       or *B*) does not have an ephemeral key, and so the above algorithm uses instead the  
1540       corresponding static key:
  - 1541           • If party *A* does not have an ephemeral key, then  $d_{e,A}$  and  $Q_{e,A}$  are respectively  
1542           instantiated by  $d_{s,A}$  and  $Q_{s,A}$ .
  - 1543           • If party *B* does not have an ephemeral key, then  $Q_{e,B}$  is instantiated by  $Q_{s,B}$ .

#### 1544 **A.4. Subcategory C1.4: Cat1 “Symmetric”**

1545 The “symmetric” subcategory includes primitives for the NIST-approved symmetric-key  
1546 enciphering scheme (the advanced encryption standard [AES]), as well as for other NIST-  
1547 approved primitives used for KDM/KC. Some primitives in scope (e.g., hashing) are techni-  
1548 cally defined as keyless, but in practice they can be considered in settings (e.g., for KDM/KC)  
1549 where their “plaintext” input is a key (symmetrically) known by two parties.

1550 While “symmetric” primitives are often used in standardized “modes of operation” for large  
1551 inputs, the thresholdization focus of this call is on the basic primitives, where the complexity  
1552 of specifying a threshold scheme lies. For example, once a threshold scheme for AES  
1553 enciphering/deciphering is defined, then it is straightforward to apply it to some mode of  
1554 operation based on AES, including for the purpose of computing a cipher-based message  
1555 authentication code (CMAC), or a ciphertext based on a mode for authentication encryption  
1556 with associated data (AEAD). Similarly, a threshold scheme for an approved hash function  
1557 could then also be applied to calculate an HMAC. Some threshold schemes may nonetheless  
1558 allow a cost amortization when repeatedly executed.

##### 1559 **A.4.1. Subcategory C1.4.1: AES Enciphering/Deciphering**

1560 With respect to threshold enciphering/deciphering in Cat1, there is only one symmetric-key  
1561 block-cipher of interest: AES, specified in FIPS-197. A submission of threshold scheme  
1562 for AES enciphering/deciphering must assume a secret-sharing of the secret key, and  
1563 should provide implementations for at least the key-sizes 128 and 256. A submission  
1564 can choose to implement any (or various) types of input/output interface from {NSS, SSI,  
1565 SSO and SSIO}. In applications where the high-sensitivity of the plaintext warrants a

distribution of trust over its knowledge, then it can make sense to consider: an SSI mode for enciphering, and/or an SSO mode for deciphering, so that the plaintext is not reconstructed within the decentralized AES-evaluator. For benchmarking purposes, a submission should evaluate performance at least in the single evaluation case, i.e., for a single AES enciphering and/or deciphering. However, to help clarify possible amortization gains and/or clarify the feasibility of the threshold approach for AES modes of operation (in the [SP800-38-series](#)), the benchmarking can also measure performance for the threshold execution of  $2^6$  and/or  $2^{10}$  AES encipherings/decipherings in some specific mode of operation.

**Threshold AES enciphering versus oblivious AES evaluation.** Oblivious AES evaluation is a common secure 2-party computation (S2PC) benchmark in the literature. There, a single party holding the plaintext does not share it with a single party holding the key, and yet receives the corresponding ciphertext. The application of threshold AES in scope in this call is different, in that the threshold entity is responsible for computing the output, when the key has been secret-shared. The plaintext is either (i) directly shared with the threshold-decentralized entity responsible for the enciphering or deciphering, or (ii) is secret-shared in the input/output. A secret-shared-I/O threshold AES enciphering may also be useful for the computation of a CMAC, which can in turn be useful for 2KE KDM/KC. That said, techniques developed for threshold AES are likely to also be useful for oblivious AES evaluation.

#### **A.4.2. Subcategory C1.4.2: KDM and KC for 2KE**

The protocols for pair-wise key-establishment (2KE), in both the ECC-based [[SP800-56A-Rev3](#)] and RSA-based [[SP800-56B-Rev2](#)] cases, are finalized with the use of a key-derivation mechanism (KDM) [[SP800-56C-Rev2](#); [SP800-108-Rev1](#)] and optional key-confirmation (KC). These operations follow after the generation of a precursor intermediate secret *M*, obtained/produced via a key-agreement of key-transport type of 2KE protocol.

**Threshold unfriendliness.** The current NIST-specified KDM and KC primitives are possible to thresholdize based on complex MPC protocols, but are based on threshold-unfriendly hash-or-XOF functions ([[FIPS-180-4](#); [FIPS-202](#)]) or MAC/PRFs (of the type CMAC [[SP800-38B](#)], HMAC [[FIPS-198-1](#)] or KMAC [[SP800-185](#)]).

Considering the “pair-wise” nature of key-establishment protocols (i.e., involving two sides), some use cases (namely when party A has to be thresholdized, but party B has to use a legacy implementation) may require the use of a KDM and/or KC that is functionally-equivalent to a currently NIST-specified one. However, the costs and benefits of implementing a potentially costly MPC in such a case should be carefully considered.



Threshold schemes for AES enciphering/deciphering may be easy to adapt to threshold schemes for CMAC primitives. Techniques used to enable threshold schemes for the hashing that is useful for KDM or KC may also be reusable for (pseudorandom) EdDSA and Deterministic ECDSA, which require a secret-nonce computed as a hash whose pre-image contains a private nonce-derivation key.

[Cat2](#) of this call enables proposals of threshold-friendlier KDM and KC primitives that would still retain the desired properties of the final generated key, namely indistinguishability from uniform selection, and one-wayness with respect to the intermediate key  $Z$  used as input.

#### A.4.2.1. Key Derivation Mechanism (KDM)

A threshold KDM scheme makes sense if the corresponding party (in the pair-wise key-establishment) is supposed to not learn the final secret  $k$ . The threshold KDM scheme produces a secret-shared output (SSO) (similar to a threshold keygen scheme), so that the final secret  $k$  (to be consumed by another primitive) is secret-shared. There are one-step (extraction) and two-step (extract-then-expand) KDMs (see [SP800-108-Rev1](#) for the second step). Additionally, there are variants (see [SP800-135-Rev1](#)) approved for specific applications.

Since the final key  $k$  can be easily derived from the intermediate key  $M$ , it follows that it only makes sense to thresholdize a KDM if the input (intermediate) key  $M$  is also secret-shared. Conversely, if a KDM is not thresholdized but  $Z$  has itself been produced in a threshold manner, (i.e., based on a secret-shared private key  $d$ ), then the reconstruction of  $Z$  does not break the confidentiality of the private key  $d$ .

#### A.4.2.2. Key Confirmation (KC)

A threshold **key-confirmation** primitive computes a PRF image of the intermediate secret  $Z$ , without  $Z$  ever being reconstructed. This can make sense if the KDM is also thresholdized in SSI mode, to directly use a secret-shared  $Z$  as input, without needing to reconstruct it. Key-confirmation is defined, in various possible modes (unilateral or bilateral), for ECC-based key-agreement in [SP800-56A-Rev3](#) (§5.9, Table 5) and RSA-based key-establishment in [SP800-56B-Rev2](#) (§5.6, Table 1).

### A.5. Subcategory C1.5: key-Generation (keygen) for Cat1 Schemes

A key-generation (keygen) primitive determines a private/secret “key” that is needed by subsequent primitives. The threshold scheme may also compute other public parameters. For

example, the keygen primitive of a digital signature scheme produces a private/public keypair, whose private element is then required to produce signatures, and whose public element is used to verify the correctness of signatures. Typical requirements for private keys include unbiasing and confidentiality. These requirements can also apply to the generation of other secret material, such as a random secret nonce. Secrets generated via a keygen primitive may be persistent (e.g., for multiple-times use, without planned erasure), or ephemeral (e.g., for single-time use, followed by erasure). Table 10 provides a non-exhaustive list of parameters that may be generated via a keygen operation (some variations are possible).

**Table 10.** Examples of keygen purposes

Keygen purpose (subsequent operation)	Private/secret key	Other public elements
ECC-signing; ECC-2KA primitives	exponent $d$ (integer mod $n$ )	$Q = d \cdot G$ (elliptic curve point)
RSA signing and decryption	primes $(p, q)$	modulus $N = p \cdot q$
	exponent $d = e^{-1} \bmod \phi_N$	exponent $e$
RSA encryption for 2KE	random bit-string $Z$	$c = \text{RSAEP}((n, e), Z)$
Key-derivation / key-confirmation		KC( $Z, \dots$ )
AES enciphering/deciphering	random bit-string $k$	—

**Terminology and scope for threshold schemes for keygen.** Threshold schemes for keygen are often called **distributed key generation (DKG)** protocols. In this call, the focus on DKG is only on the generation of the private/secret keys and (when applicable) the public parameters that depend on them (e.g., an RSA modulus obtained from the product of two secret primes, or the elliptic curve public point obtained from integer-multiplying a base point by the secret key). Other “domain parameters”, such as the security strength  $\kappa$ , the parameters of an elliptic curve, or an RSA encryption key, which may be determined before the computation of the private key (but which in conventional specifications may sometimes be included within the keygen primitive) can be assumed to be fixed or pre-agreed upon.

**Interchangeability of random values.** In a DKG protocol, the random private/secret key to be output in secret-shared form, and possibly other intermediate random elements, is obtained by combining random contributions from several parties. This call does not pose specific requirements on these random values, i.e., beyond the requirement of **interchangeability** with regard to some subsequent operation of interest. However, a submitted DKG protocol should be accompanied by an explanation of why the proposed randomness generation mechanism provides appropriate security assurances, namely compared to the



assurances provided by the conventional random-bit generation (RBG) [SP800-90A-R1; SP800-90B; SP800-90C-3PD] that may be required in the corresponding conventional (non-threshold) keygen specification. Some original RBG-related requirements associated with random values in the conventional specification may still be considered for the individual contributions of each party in a corresponding DKG.

#### A.5.1. Subcategory C1.5.1: ECC Keygen (for ECDSA, EdDSA, and ECC-2KA)

The ECC keygen of a private/public key-pair is similar across various schemes, including for ECDSA and EdDSA signature schemes [FIPS-186-5-Draft], and for ECC-2KA primitives, such as CDH and MQV [SP800-56A-Rev3]. In a threshold **ECC keygen** (i.e., DKG for an ECC scheme), the usual goal is to produce a secret-sharing  $[d]$  of a private key  $d$  (usually a positive integer mod  $n$ , the order of the subgroup of interest), along with a corresponding (not-secret-shared) public key  $Q = d \cdot G$ . In a threshold 2KA scheme, each party may need this decentralization (secret-sharing) for their static private key  $d_A$  (or  $d_{s,A}$ ) and/or an ephemeral private key ( $d_{e,A}$ ).

Some schemes, such as EdDSA, may include additional private/secret elements (e.g., a nonce-derivation key for pseudorandom generation of nonces) that do not require a subsequent verifiable relation with the public key. The generation of said components in the threshold setting may be considered differently (or may even not be necessary), provided that an appropriate **interchangeability** property is satisfied with regard to the subsequent operations that use the ECC private/public keypair.

Submissions of threshold schemes for ECC signing and ECC-2KA primitives are expected (though not required) to include a corresponding proposal of a compatible ECC-DKG protocol. Implementation recommendations for a submitted DKG (e.g., which elliptic curves and security parameters) should apply to at least one subsequent threshold scheme of interest.

#### A.5.2. Subcategory C1.5.2: RSA Keygen

RSA keygen is needed for the RSADSA scheme (Section A.1.1) and the RSA PKE scheme used for 2KE (Section A.2). In its *basic* format, RSA keygen consists of:

- generating a pair of random secret primes  $(p, q)$ , and outputting their product  $N$ ; and
- computing and outputting as private key  $d$  the inverse (mod  $\text{LCM}(p-1, q-1)$ ) of a public exponent  $e$ , where  $e$  is selected (randomly or as an input parameter) before the selection of the primes.

DKG schemes for RSA can be submitted separately from subsequent threshold operations, such as threshold RSA signing, threshold RSA decryption, or threshold RSA SSI-encryption. Still, a submission of RSA DKG should be compatible with said subsequent schemes, and should include evaluation for at least two security parameters consistent with the recommendations from Table 5.

FIPS-186-5-Draft (§A.1) and SP800-56B-Rev2 (§6.2–§6.3) specify various requirements for the RSA keygen, respectively for signing and PKE. Possible variations of the format of the output key include the *prime-factor* format and the *CRT* format, as explained in Section A.2.2. The following paragraph list some of the requirements.

#### A.5.2.1. Criteria for the RSA Modulus and Primes

- $p$  and  $q$  must be of the same bit length (i.e., half the length of the RSA modulus  $N$ ).
- $p$  and  $q$  must be randomly generated (but the two most significant bits of each may be arbitrarily set), as “probable” or “provable” primes, satisfying at least one of the five options from Table 11.

**Table 11.** Criteria for the random primes of an RSA modulus

Type	Sub-type	Provable prime	Probable prime
Simple	provable	$p, q$	
	probable		$p, q$
Complex	provable	$p_1, p_2, q_1, q_2$	$p, q$
	hybrid	$p_1, p_2, q_1, q_2,$	$p, q$
	probable		$p_1, p_2, q_1, q_2, p, q$

Per §A.1.1 of FIPS-186-5-Draft:  $p_1, p_2, q_1, q_2$  are called auxiliary primes and must be divisors of  $p-1, p+1, q-1$  and  $q+1$ , respectively, i.e.,  $p_1|p-1, p_2|p+1, q_1|q-1, q_2|q+1$ .

To satisfy the “complex” type of key-generation, the auxiliary primes must exist with certain minimum lengths. If  $p$  and  $q$  are required to be provable primes, then their minimal required bit-length is roughly half of the minimal required length of probable primes.

In a submitted RSA DKG, the threshold computation of the primes and modulus may be modularized from the subsequent calculation of the private decryption/signing exponent  $d$ . Interestingly, there are conceivable applications (beyond signatures, encryption, and decryption) where RSA moduli are useful and a private exponent is not necessary.

1722 **A.5.2.2. Criteria for the Private Exponent**

1723 The private exponent  $d = e^{-1} \pmod{L}$ , where  $L = \text{LCM}(p-1, q-1)$ , must be larger than  
1724  $2^{nlen/2}$  and smaller than  $L$ , where the public exponent  $e$  is an integer between  $2^{16}$  and  $2^{256}$   
1725 selected before the generation of  $p$  and  $q$ .

1726 **A.5.3. Subcategory C1.5.3: Bitstring Keygen**

1727 Various primitives require the random generation of a secret bit-string (or integer within a  
1728 defined interval), without the need for a corresponding public component. For example, this  
1729 is the case with generating: an AES key; a secret-key for encapsulation under an RSA PKE;  
1730 a nonce for use in other schemes; a salt for a KDM or KC in the scope of a 2KA.

1731 A DKG based on verifiable secret-sharing may require public commitments of the shares of  
1732 each party, even if the original primitive did not require any public key. A submission should  
1733 explain how/whether the cryptographic assumptions sustaining the security of the threshold  
1734 scheme change in comparison with those required for the security of the original primitive.  
1735 For example, AES-256 is considered to be post-quantum secure, whereas ECC-based  
1736 commitments used in typical MPC protocols might not be.

1737 **A.6. Subcategory C2.6: Advanced**

1738 As mentioned in Section 7.2.1, subcategory C2.6 allows for the submission of threshold  
1739 schemes for primitives that support cryptographic schemes with advanced functional features  
1740 that are different from those in current NIST standards. For example, in the case of a  
1741 fully-homomorphic encryption (FHE) scheme, the supported operations go beyond the usual  
1742 keygen, encryption and decryption from a regular encryption scheme. There is also a set of  
1743 homomorphic operations (e.g., addition and multiplication) over ciphertexts (see, e.g., [HES,  
1744 §1.1.1]). As another example, an identity-based encryption (IBE) scheme has not just one  
1745 key-generation primitive, but rather two: one for generating a public key and a master private  
1746 key, and another one (requiring the master key as input) for generating a decryption key for  
1747 each possible “identity” (e.g., email addresses). A generalization of IBE is attribute-based  
1748 encryption (ABE), where the private key of each user is created based on a set of attributes.

1749 In this subcategory, the selection of the use-cases used to benchmark performance is left to  
1750 the discretion of the submitters. For example, different FHE schemes may require different  
1751 benchmarking operations to highlight their best features. One FHE scheme may be better  
1752 suited to homomorphic Boolean operations (operations over bits), while another one may be  
1753 better suited for homomorphic modular operations over large integers.

1754 **A.6.1. Use-Case Example: Non-Threshold FHE-Based AES Oblivious Enciphering**

1755 0a. **Setup FHE (keygen):** An FHE scheme is initialized with encryption key  $e$  (for encryp-  
1756 tion operation  $\text{FHE.Enc}_e$ ), and decryption key  $d$  (for decryption operation  $\text{FHE.Dec}_d$ ),  
1757 and allows homomorphic-evaluation (over FHE-ciphertexts) of any function  $f$  (within  
1758 a certain range of functions) using operation  $\text{FHE.Hom}[f]$ .

1759 0b. **Setup AES (keygen):** An AES cipher is initialized with secret key  $k$ , with  $\text{AES.Enc}_k$   
1760 denoting the corresponding enciphering operation.

1761 0c. **Setup parties (private inputs):** (i) Client  $A$  knows a secret plaintext  $m$ , and the FHE  
1762 encryption key  $e$ ; (ii) Server  $S$  knows the AES secret-key  $k$ ; (iii) and client  $B$  (possibly  
1763 the same as client  $A$ ) knows the FHE decryption key  $d$ .

1764 1. **FHE-Encrypt.** The client  $A$  FHE-encrypts the secret plaintext  $m$ , obtains the FHE-  
1765 ciphertext  $C = \text{FHE.Enc}_e(m)$ , and sends it to the server  $S$ .

1766 2. **FHE-Homomorphic-Evaluate.** The server  $S$  homomorphically evaluates the AES-  
1767 enciphering, obtains  $H = \text{FHE.Hom}[\text{AES.Enc}_k](C)$  (which is a valid FHE-encryption  
1768 of the AES-enciphering of secret plaintext  $m$ ), and sends the result to client  $B$ .

1769 3. **FHE-Decrypt.** The client  $B$  FHE-decrypts the received ciphertext  $H$ , and thus obtains  
1770 the AES-enciphering of the secret plaintext:  $\text{AES.Enc}_k(m) = \text{FHE.Dec}_d(H)$ .

1771 4a. **(Optional) Prove correctness.** The server  $S$  may also send a ZKPoK string  $\pi =$   
1772  $\text{ZKPoK.Prove}[k; (H, C) : \text{FHE.Hom}[\text{AES.Enc}_k](C) = H]$  to client  $B$ , thus ZK-proving  
1773 knowledge of a secret AES key ( $k$ ) that is consistent with the homomorphic operation  
1774 that transformed the initial FHE-ciphertext  $C$  into the final FHE-ciphertext  $H$ . A more  
1775 sophisticated ZKPoK can also be used to prove consistency with some additional  
1776 public commitment of the AES-key  $k$ .

1777 4b. **Verify the proof.** Anyone with the FHE-ciphertexts  $(C, H)$  can verify the correctness  
1778 of the ZKPoK  $\pi$ , by checking  $\text{true} \stackrel{?}{=} \text{ZKPoK.Verify}(\pi, (H, C), \text{AES.Enc})$ .

1779 **External engagement.** Proposals of FHE schemes (and their threshold schemes) are  
1780 welcome to be submitted and/or analyzed in connection with other related ongoing public  
1781 efforts, such as HomomorphicEncryption.org and FHE.org, as a way of promoting: (i)  
1782 fulfillment of community-based technical recommendations; (ii) alignment with existing  
1783 reference material/specifications; and (iii) further public scrutiny of proposed schemes. Such  
1784 engagements may also help clarify reference use-cases for useful benchmarking.

## 1785 **A.6.2. Threshold Schemes for FHE-based AES Oblivious Enciphering**

1786 Once a conventional (non-threshold) scheme is specified (S6) in scope of the “advanced”  
1787 subcategory C2.6, there may be multiple types of decentralization to consider. For the above-  
1788 described example of FHE application (Section A.6.1), the following is a non-exhaustive list  
1789 of possible decentralizations of one of the original participants (client *A*, server *S*, or client  
1790 *B*) into a threshold entity composed of multiple parties.

- 1791 1. **Threshold FHE.Keygen.** In a setup phase with a thresholdized client *B*, a DKG can  
1792 distributively compute a secret-sharing of an FHE decryption key  $d$ . Whether or not  
1793 the encryption key  $e$  is secret-shared can depend on whether the FHE scheme is of,  
1794 respectively, symmetric-key or asymmetric-key (i.e., public/private key pair) type.
- 1795 2. **SSI threshold FHE-Encryption.** If client *A* is thresholdized, and set up with a secret-  
1796 shared plaintext  $m$ , a threshold scheme can compute  $C = \text{FHE.Enc}_e(m)$  without  
1797 anyone learning  $m$ .
- 1798 3. **Threshold Homomorphic evaluation (of function with secret parameter).** If the  
1799 server *S* is thresholdized, and setup with a secret-sharing of the AES key  $k$ , then the  
1800 parties can distributively compute the homomorphic-evaluation operation, to obtain  
1801  $H = \text{FHE.Hom}[\text{AES.Enc}_k](C)$ , without anyone learning  $k$ .
  - 1802 • In an NSS mode, all server-parties learn  $H$ .
  - 1803 • In an SSO mode, each server learns a secret-share of  $H$ .
- 1804 4. **Threshold FHE decryption.** If client *B* is thresholdized, and setup with a secret-  
1805 sharing of the FHE-decryption key  $d$ , then a threshold scheme can decrypt the received  
1806 value  $H$  to obtain  $C = \text{AES}_k(m)$ , without anyone learning  $d$ .
  - 1807 • In a NSS mode, all clientB-parties learn  $C$ .
  - 1808 • In a SSO mode, each clientB-party learns only a secret-share of  $C$ .
- 1809 5. **Threshold ZKPoK.** (See subcategory C2.7 in Section A.7)

1810 **On the use case of oblivious AES enciphering.** The use case is called oblivious AES-  
1811 enciphering because the client *B* obtained an AES-enciphering of the secret plaintext  $m$   
1812 even though the AES-key holder (the server *S*) remained oblivious to the secret plaintext.  
1813 Interestingly, oblivious AES-enciphering is also a typical benchmark case for secure 2-party  
1814 computation (S2PC; consider the case where clients *A* and *B* are the same), usually using  
1815 different techniques, such as garbled circuits and/or oblivious transfer. Compared with an  
1816 FHE-based solution, usual S2PC protocols (expectably) lead to much faster execution, but  
1817 also much larger communication complexity.

## 1818 **A.7. Subcategory C2.7: ZKPoKs**

1819 Besides (secure) **multi-party computation** (MPC), a broad type of primitive of great interest  
1820 in the threshold context is the **zero-knowledge proof of knowledge** (ZKPoK), which is  
1821 covered by subcategory C2.7. As mentioned in Section 7.2.2, a submission of ZKPoK in  
1822 this subcategory must specify a conventional ZKPoK, and possibly also specify a threshold  
1823 version (when the prover is distributed and there is a secret-sharing of the secret input).

1824 In usual ZKP terminology [ZkpComRef], a ZKPoK is used to prove a **statement** of knowledge,  
1825 such as knowledge of a secret **witness** ( $w$ ) that satisfies a given **relation** ( $R$ ) with a public  
1826 **instance** ( $x$ ), such that  $R(x, w)$  is true. For example, in a ZKPoK of a private RSA key, the  
1827 *instance* can be the RSA modulus  $N$ , the secret *witness* can be the corresponding pair  $(p, q)$   
1828 of prime factors, and the *relation* can be the predicate that returns true if and only if the  
1829 input witness is indeed a pair of primes and their product is the public modulus.

### 1830 **Type of “proofs” of interest:**

- 1831 • **Proofs and arguments:** The use of “proof” in this call is meant to also include the  
1832 case of *arguments* with computational soundness. Any submission of ZKPoK should  
1833 clarify its soundness type (to allow for differentiation between “proof” and argument).
- 1834 • **ZKP of knowledge (versus of correctness):** The proofs in scope are ZKPoKs, but can  
1835 also serve the purpose of ZK-proving *correctness* of the secret data (whose knowledge  
1836 is being proven) as well as of the corresponding public data. In the literature, a ZKP  
1837 of correctness is also known as a ZKP of “language membership”.
- 1838 • **Transferable and non-interactive.** Traditionally, ZKPs and ZKPoKs are defined as  
1839 two-party protocols with a requirement of deniability (also known as non-transferabil-  
1840 ity), implying that a verifier convinced by a proof cannot later transfer said confidence  
1841 to a third party. This property often stems from interactivity between prover and  
1842 verifier, and/or relies on local setup assumptions, such as a local common reference  
1843 string (CRS) or local random oracle (RO). Conversely, the present call is by default  
1844 interested on transferable non-interactive zero-knowledge (NIZK) proofs that can be  
1845 publicly verified non-interactively. A submission of ZKPoK can deviate from this  
1846 default (non-interactiveness and transferability) as long as justified on the basis of  
1847 utility to the threshold setting.

1848 The instantiation of some of the above-listed attributes (e.g., transferability, and compu-  
1849 tational soundness) may affect some aspects of composability. These effects should be  
1850 discussed in any submission that proposes a ZKPoK.

**Distributed prover (not verifier).** In this call, the default setting of interest for thresholdization of a ZKPoK is the secret-sharing, across multiple parties, of the secret key (traditionally held by a single prover) whose knowledge is being proven. While a ZKPoK variant can also be conceived for the case of distributed verification (with the ZK property requiring that a threshold number of verifier parties do not collude), such setting is not the default. A deviation from the mentioned default in a submission of ZKPoK is possible but its auxiliary utility for the threshold setting then needs to be thoroughly argued for.

**Examples.** Table 12 lists various examples of ZKPoK of anticipated interest with regard to Cat1 primitives. Other examples can be conceived for primitives in Cat2.

**Table 12.** Example ZKPoKs of interest related to Cat1 primitives

Related type	Related (sub)sub-category: Primitive	Example ZKPoK (including consistency with public commitments of secret-shares, when applicable)
Keygen	C1.5.1: ECC keygen	of discrete-log ( $s$ or $d$ ) of pub key $Q$
	C1.5.2: RSA keygen	of factors ( $p, q$ ), or group order $\phi$ , or decryption key $d$
	C1.5.3: AES keygen	of secret key $k$ (with regard to secret-sharing commitments)
PKE	C1.2.1: RSA encryption	of secret plaintext $m$ (encrypted)
	C1.2.2: RSA decryption	of secret-shared plaintext $m$ (after SSO-threshold decryption)
Symmetric	C1.4.1: AES enciphering	of secret key $k$ (with regard to plaintext/ciphertext pair)
	C1.4.2: Hashing in KDM	of secret pre-image $Z$

Some observations:

- A ZKPoK of a secret AES key that transforms a given plaintext into a given ciphertext corresponds to a signature primitive submitted to the PQC process.
- No ZKPoK example was provided in association with the signing operation, since their public verification operation already inherently verifies the signature correctness. In fact, a digital signature often constitutes a transferable NIZKPoK of the private signing key corresponding to the public key, with said proof being additionally bound to a message (the element being signed). For example, an EdDSA/Schnorr signature (Section A.1.1) is itself a NIZKPoK of discrete-log.
- The cases of ZKPoK related to a private **signing** key, but possibly without producing a signature, are associated with keygen (subcategories C1.5 and C2.5).

If a submission of threshold scheme uses a ZKP/ZKPoK that may be of interest to support other threshold schemes, then it should modularize the specification of said ZKP/ZKPoK and indicate it as useful also for consideration in subcategory C2.7.



1883 **Submission of a ZKPoK as auxiliary to other threshold scheme(s):**

- 1884 • **Specification of a non-threshold version.** A submission in the ZKPoK subcategory  
1885 must specify a conventional (non-threshold) ZKPoK. This may be submitted without  
1886 a corresponding distributed/threshold version, as long as the documentation clarifies  
1887 how the conventional ZKPoK can be useful for the threshold setting (perhaps some  
1888 other concrete threshold scheme). For example, a conventional ZKPoK can be justified  
1889 for use by a dealer to prove correctness of an established secret-sharing setup. There  
1890 may nonetheless be an additional value in also specifying a threshold version of the  
1891 ZKPoK (i.e., when the secret input is distributed).
- 1892 • **Standalone versus embedded proposal of a ZKPoK.** A package that proposes  
1893 an auxiliary ZKPoK (and possibly a distributed version thereof) can be submitted  
1894 within the standalone ZKPoK subcategory, or within a submission of a threshold  
1895 scheme(s) for other primitives in [Cat1](#) or [Cat2](#). In the standalone case, the proposal  
1896 must clarify how the secret and public knowledge matches the setting of (e.g., a  
1897 particular secret-sharing useful for) a threshold scheme for some primitive of interest.
- 1898 • **External engagement.** Proposals of ZKPoK schemes (and their threshold schemes)  
1899 are welcome to be submitted and/or analyzed in connection with other related on-  
1900 going public efforts, such as ZKProof.org, as a way of promoting: (i) fulfillment of  
1901 community-based technical recommendations; (ii) alignment with existing reference  
1902 material/specifications; and (iii) further public scrutiny of proposed schemes. Such  
1903 engagements may also help clarify reference use-cases for useful benchmarking.

1904 **Notes on features.**

- 1905 • **Succinctness:** For practicality, **succinctness** is a useful feature of a ZKPoK. When  
1906 focusing on succinct and non-interactive ZKPoKs, it is also common to refer to them  
1907 as SNARKs (succinct **n**on-interactive **a**rguments of **k**nowledge).
- 1908 • **Transferability:** As mentioned [above](#), non-interactive public verifiability / transfer-  
1909 ability are default desired features
- 1910 • **Security assumptions:** While the assessment of security of a ZKPoK may be based on  
1911 assumptions different from those inherent to the underlying cryptographic primitive,  
1912 or to a related proposed threshold scheme, said implications should be distinguished  
1913 across various security properties. In particular, it is relevant to characterize the  
1914 properties of ZK, soundness and non-malleability, and how they may vary upon  
1915 various types of protocol composition (e.g., concurrent executions).



**Specialized versus generic ZKPoKs.** Some ZKPoKs (e.g., of a discrete-log, or of an RSA private key) may be based on specialized techniques somewhat similar to the operations (e.g., exponentiations) used to commit the secret pre-image. Conversely, other ZKPoKs (e.g., when proving knowledge of a pre-image of AES-enciphering, or of SHA-based hashing) may stem more easily from a generic ZKP system that simply requires “arithmetizing” the *statement* of knowledge, the *instance* and the *witness* in some suitable representation (e.g., specifying a Boolean or arithmetic circuit, and instantiating its input variables). In the latter case, a submitted ZKPoK can be explained generically, and then a simple explanation be given on how to apply it to a circuit (or other applicable representation). For example, the NIST Circuit Complexity project [Proj-CC] collects Boolean circuit representations of various NIST-approved primitives, such as from AES and SHA. The final version of this call may reference a specific representation for Boolean circuits, to facilitate an interchangeable specification of circuits of certain NIST-specified primitives (e.g., of certain block-ciphers and hash-functions) whose proof of knowledge of pre-image may be useful.

#### **A.8. Subcategory C2.8: (Auxiliary) Gadgets**

As mentioned in Section 7.2.3, subcategory C2.8 allows for the consideration of gadgets, such as garbled circuits, oblivious transfer, generation of correlated randomness, commitments, secret resharing (possibly for a new threshold value and a new total number of parties), multiplicative-to-additive share conversion, additively homomorphic encryption (AHE), MPC or ZKP friendly hashing, consensus, and broadcast. The specification of some gadgets may also fit other subcategories. For example, an AHE scheme allows for an *advanced* feature (homomorphic addition over ciphertexts), and thus can fit in “advanced” subcategory C2.6 (if accompanied by a corresponding threshold scheme), and at the same time can also be useful to support multiple other threshold schemes, and thus fit in subcategory C2.8. In such type of cases, a submission should identify (e.g., including in S2 and S3) the fit in various subcategories.

Gadgets can be proposed in a standalone manner in a submission, or as a module in a more encompassing submission in the scope of other subcategories. A standalone submission of an auxiliary gadget (and possible threshold version thereof) should make a strong case for its utility in supporting the threshold environment, and/or in directly supporting various concrete threshold schemes in scope of other subcategories in this call.

## 1947 **B. Submission Checklists**

1948 The following are draft templates of checklists to help keep track of the fulfillment of the  
1949 various requirements for a complete submission:

### 1950 **B.1. Checklist for Submission Phases (Ph) (see Section 4)**

1951	Check	#	Item	Comments
1952	<input type="checkbox"/>	<a href="#">Ph1</a>	(Optional) Early abstract	
1953	<input type="checkbox"/>	<a href="#">Ph2</a>	(Optional) Preliminary package	
1954	<input type="checkbox"/>	<a href="#">Ph3</a>	Full package ( <a href="#">M1–M5</a> )	

### 1955 **B.2. Checklist for Package Main Components (M) (see Section 4)**

1956	Check	#	Item	Comments
1957	<input type="checkbox"/>	<a href="#">M1</a>	Written specification ( <a href="#">S1–S16</a> )	
1958	<input type="checkbox"/>	<a href="#">M2</a>	Reference implementation ( <a href="#">Src1–Src4</a> )	
1959	<input type="checkbox"/>	<a href="#">M3</a>	Execution instructions ( <a href="#">X1–X7</a> )	
1960	<input type="checkbox"/>	<a href="#">M4</a>	Experimental evaluation ( <a href="#">Perf1–Perf5</a> )	
1961	<input type="checkbox"/>	<a href="#">M5</a>	Additional statements	

### 1962 **B.3. Checklist for M1: Written Specification Sections (S) (see Section 4.2)**

1963	Check	#	Item	Comments
1964	<input type="checkbox"/>	<a href="#">S1</a>	Title pages	
1965	<input type="checkbox"/>	<a href="#">S2</a>	Abstract	
1966	<input type="checkbox"/>	<a href="#">S3</a>	Executive summary	
1967	<input type="checkbox"/>	<a href="#">S4</a>	Index	
1968	<input type="checkbox"/>	<a href="#">S5</a>	Clarification of prior work	
1969	<input type="checkbox"/>	<a href="#">S6</a>	Conventional primitives/scheme	
1970	<input type="checkbox"/>	<a href="#">S7</a>	System model	
1971	<input type="checkbox"/>	<a href="#">S8</a>	Protocol description	
1972	<input type="checkbox"/>	<a href="#">S9</a>	Security analysis	
1973	<input type="checkbox"/>	<a href="#">S10</a>	Analytic complexity	
1974	<input type="checkbox"/>	<a href="#">S11</a>	Choices and comparisons	
1975	<input type="checkbox"/>	<a href="#">S12</a>	Technical criteria	
1976	<input type="checkbox"/>	<a href="#">S13</a>	Deployment recommendations	
1977	<input type="checkbox"/>	<a href="#">S14</a>	Notation	
1978	<input type="checkbox"/>	<a href="#">S15</a>	References	
1979	<input type="checkbox"/>	<a href="#">S16</a>	Appendices (optional)	

**1980 B.4. Checklist for M2: Open source (Src) Reference Implementation (see Section 4.3)**

1981	Check	#	Item	Comments
1982	<input type="checkbox"/>	<a href="#">Src1</a>	Is self-contained	
1983	<input type="checkbox"/>	<a href="#">Src2</a>	Is licensed as open-source	
1984	<input type="checkbox"/>	<a href="#">Src3</a>	Contains inline comments	
1985	<input type="checkbox"/>	<a href="#">Src4</a>	Has a clear API	

**1986 B.5. Checklist for M3: Execution Instructions (X) (see Section 4.4)**

1987	Check	#	Item	Comments
1988	<input type="checkbox"/>	<a href="#">X1</a>	User manual: compilation	
1989	<input type="checkbox"/>	<a href="#">X2</a>	User manual: parametrization	
1990	<input type="checkbox"/>	<a href="#">X3</a>	User manual: execution	
1991	<input type="checkbox"/>	<a href="#">X4</a>	User manual: KAT set	
1992	<input type="checkbox"/>	<a href="#">X5</a>	Script: KAT	
1993	<input type="checkbox"/>	<a href="#">X6</a>	Script: benchmark	
1994	<input type="checkbox"/>	<a href="#">X7</a>	Script: others (optional)	

**1995 B.6. Checklist for M4: Performance Analysis (Perf) (see Section 4.5)**

1996	Check	#	Item	Comments
1997	<input type="checkbox"/>	<a href="#">Perf1</a>	Memory complexity	
1998	<input type="checkbox"/>	<a href="#">Perf2</a>	Processing time	
1999	<input type="checkbox"/>	<a href="#">Perf4</a>	Networking time	
2000	<input type="checkbox"/>	<a href="#">Perf3</a>	Communication complexity	
2001	<input type="checkbox"/>	<a href="#">Perf5</a>	Round complexity	

**2002 B.7. Checklist for Technical Requirements (T) (see Section 5)**

2003	Check	#	Item	Comments
2004	<input type="checkbox"/>	<a href="#">T1</a>	Primitives	
2005	<input type="checkbox"/>	<a href="#">T2</a>	System model	
2006	<input type="checkbox"/>	<a href="#">T2.1</a>	Participants	
2007	<input type="checkbox"/>	<a href="#">T2.2</a>	Distributed systems and communication	
2008	<input type="checkbox"/>	<a href="#">T2.3</a>	Adversary	
2009	<input type="checkbox"/>	<a href="#">T3</a>	Security idealization	
2010	<input type="checkbox"/>	<a href="#">T4</a>	Security versus adversaries	
2011	<input type="checkbox"/>	<a href="#">T4.1</a>	Active	
2012	<input type="checkbox"/>	<a href="#">T4.2</a>	Adaptive	
2013	<input type="checkbox"/>	<a href="#">T4.3</a>	Pro-active	
2014	<input type="checkbox"/>	<a href="#">T5</a>	Threshold profiles	
2015	<input type="checkbox"/>	<a href="#">T6</a>	Building blocks	

## 2016 References

- 2017 [FIPS-180-4] National Institute of Standards and Technology (NIST). *Secure Hash Standard (SHS)*.  
2018 (U.S. Department of Commerce) Draft Federal Information Processing Standards Publication  
2019 (FIPS PUBS) 180-4. August 2015. DOI: [10.6028/NIST.FIPS.180-4](https://doi.org/10.6028/NIST.FIPS.180-4).
- 2020 [FIPS-186-5-Draft] National Institute of Standards and Technology (NIST). *Digital Signature Stan-*  
2021 *dard (DSS)*. (U.S. Department of Commerce) Draft Federal Information Processing Standards  
2022 Publication (FIPS PUBS) 186-5. October 2019. DOI: [10.6028/NIST.FIPS.186-5-Draft](https://doi.org/10.6028/NIST.FIPS.186-5-Draft).
- 2023 [FIPS-197] National Institute of Standards and Technology (NIST). *Advanced Encryption Standard*  
2024 *(AES)*. *Federal Information Processing Standards Publication 197*. November 2001. DOI:  
2025 [10.6028/NIST.FIPS.197](https://doi.org/10.6028/NIST.FIPS.197).
- 2026 [FIPS-198-1] National Institute of Standards and Technology (NIST). *The Keyed-Hash Message*  
2027 *Authentication Code (HMAC)*. *Federal Information Processing Standards Publication 198-1*.  
2028 July 2008. DOI: [10.6028/NIST.FIPS.198-1](https://doi.org/10.6028/NIST.FIPS.198-1).
- 2029 [FIPS-202] National Institute of Standards and Technology (NIST). *SHA-3 Standard: Permutation-*  
2030 *Based Hash and Extendable-Output Functions*. August 2015. DOI: [10.6028/NIST.FIPS.202](https://doi.org/10.6028/NIST.FIPS.202).
- 2031 [HES] Martin Albrecht, Melissa Chase, Hao Chen, Jintai Ding, Shafi Goldwasser, Sergey Gorbunov,  
2032 Shai Halevi, Jeffrey Hoffstein, Kim Laine, Kristin Lauter, Satya Lokam, Daniele Micciancio,  
2033 Dustin Moody, Travis Morrison, Amit Sahai, and Vinod Vaikuntanathan. *Homomorphic*  
2034 *Encryption Standard*. Published by HomomorphicEncryption.org. November 2018. Updated  
2035 versions at <https://homomorphicencryption.org/standard/>.
- 2036 [IG-FIPS-140-2] National Institute of Standards and Technology (NIST) and Canadian Centre  
2037 for Cyber Security (CCCS). *Implementation Guidance for FIPS PUB 140-2 and the Crypto-*  
2038 *graphic Module Validation Program*. Version updated by the Cryptographic Module Validation  
2039 Program on 2022-October-07. [https://csrc.nist.gov/csrc/media/projects/cryptographic-module-](https://csrc.nist.gov/csrc/media/projects/cryptographic-module-validation-program/documents/fips140-2/FIPS1402IG.pdf)  
2040 [validation-program/documents/fips140-2/FIPS1402IG.pdf](https://csrc.nist.gov/csrc/media/projects/cryptographic-module-validation-program/documents/fips140-2/FIPS1402IG.pdf). October 2022.
- 2041 [ITL-Patent-Policy] National Institute of Standards and Technology (NIST). *Information Technology*  
2042 *Laboratory (ITL) Patent Policy*. <https://www.nist.gov/itl/publications-0/itl-patent-policy-inclu>  
2043 [sion-patents-itl-publications](https://www.nist.gov/itl/publications-0/itl-patent-policy-inclusion-patents-itl-publications). January 2019.
- 2044 [MPTC-Call2021a] Luís Brandão. (National Institute of Standards and Technology (NIST) Multi-  
2045 Party Threshold-Cryptography (MPTC)) Call 2021a for Feedback on Criteria for Threshold  
2046 Schemes. July 2021. URL: <https://csrc.nist.gov/csrc/media/projects/threshold-cryptography/do>

- 2047 [cuments/MPTC-call2021a-feedback.pdf](#). Public comments: <https://csrc.nist.gov/csrc/media>  
2048 [/projects/threshold-cryptography/documents/MPTC-Call2021a-Feedback-compilation.pdf](#).  
2049 Accessed via the NIST Computer Security Resource Center (CSRC) in January 2023.
- 2050 [MPTS] National Institute of Standards and Technology. *NIST Workshop on Multi-Party Threshold*  
2051 *Schemes 2020*. Virtual conference. November 2020. URL: <https://csrc.nist.gov/events/2020/m>  
2052 [pts2020](#).
- 2053 [NIST-IR8214A] Luís T. A. N. Brandão, Michael Davidson, and Apostol Vassilev. *NIST Roadmap*  
2054 *Toward Criteria for Threshold Schemes for Cryptographic Primitives*. (National Institute of  
2055 Standards and Technology (NIST) Internal Report) NISTIR 8214A. July 2020. DOI: [10.6028](https://doi.org/10.6028/NIST.IR.8214A)  
2056 [/NIST.IR.8214A](#). Public comments: <https://csrc.nist.gov/publications/detail/nistir/8214a/final>.
- 2057 [NIST-IR8214B-ipd] Luís T. A. N. Brandão and Michael Davidson. *Notes on Threshold EdDSA/-*  
2058 *Schnorr Signatures*. (National Institute of Standards and Technology (NIST) Internal Report)  
2059 NISTIR 8214B ipd (initial public draft). August 2022. DOI: [10.6028/NIST.IR.8214B.ipd](https://doi.org/10.6028/NIST.IR.8214B.ipd).
- 2060 [Proj-CC] National Institute of Standards and Technology. *NIST Project on Circuit Complexity (CC)*.  
2061 <https://csrc.nist.gov/projects/circuit-complexity>. See list of circuits at <https://csrc.nist.gov/projects/circuit-complexity/list-of-circuits>, and repository of circuits at [GitHub:usnistgov/Circuits](https://github.com/usnistgov/Circuits).  
2062 Accessed in January 2023.
- 2064 [Proj-LWC] National Institute of Standards and Technology. *NIST Project on Lightweight Cryptog-*  
2065 *raphy (LWC)*. <https://csrc.nist.gov/projects/lightweight-cryptography>. Accessed in January  
2066 2023.
- 2067 [Proj-MPTC] National Institute of Standards and Technology. *NIST Project on Multi-Party Thresh-*  
2068 *old Cryptography (MPTC)*. <https://csrc.nist.gov/projects/threshold-cryptography>. Accessed in  
2069 January 2023.
- 2070 [Proj-PEC] National Institute of Standards and Technology. *NIST Project on Privacy-Enhancing*  
2071 *Cryptography (PQC)*. <https://csrc.nist.gov/projects/pec>. Accessed in January 2023.
- 2072 [Proj-PQC] National Institute of Standards and Technology. *NIST Project on Post-quantum Cryptog-*  
2073 *raphy (PQC)*. <https://csrc.nist.gov/projects/post-quantum-cryptography>. Accessed in January  
2074 2023.
- 2075 [RFC7748] Adam Langley and Mike Hamburg and Sean Turner. *Elliptic Curves for Security*.  
2076 Internet Research Task Force (IRTF) Request for Comments: RFC 7748. January 2016. DOI:  
2077 [10.17487/RFC7748](https://doi.org/10.17487/RFC7748).

- 2078 [RFC8017] Kathleen Moriarty and Burt Kaliski and Jakob Jonsson and Andreas Rusch. *PKCS*  
2079 *#1: RSA Cryptography Specifications Version 2.2*. Internet Engineering Task Force (IETF)  
2080 Request for Comments: RFC 8017. November 2016. DOI: [10.17487/RFC8017](https://doi.org/10.17487/RFC8017).
- 2081 [SP800-38-series] Morris Dworkin. *Recommendation for Block Cipher Modes of Operation — Parts*  
2082 *A through G*. (National Institute of Standards and Technology) NIST Special Publications (SP):  
2083 [800-38A](#) (*Methods and Techniques*, 2021); [800-38B](#) (*the CMAC Mode for Authentication*  
2084 *2016*); [800-38C](#) (*the CCM Mode for Authentication and Confidentiality*, 2007); [800-38D](#)  
2085 (*Galois/Counter Mode (GCM) and GMAC*, 2007); [800-38E](#) (*the XTS-AES Mode for Confi-*  
2086 *dentiality on Storage Devices*, 2010); [800-38F](#) (*Methods for Key Wrapping*, 2012); [800-38G](#)  
2087 (*Methods for Format-Preserving Encryption*, 2016). October 2016.
- 2088 [SP800-38B] Morris Dworkin. *Recommendation for Block Cipher Modes of Operation: The CMAC*  
2089 *Mode for Authentication*. (National Institute of Standards and Technology) NIST Special  
2090 Publication (SP) 800-38B. May 2005. DOI: [10.6028/NIST.SP.800-38B](https://doi.org/10.6028/NIST.SP.800-38B).
- 2091 [SP800-38F] Morris Dworkin. *Recommendation for Block Cipher Modes of Operation: Methods*  
2092 *for Key Wrapping*. (National Institute of Standards and Technology) NIST Special Publication  
2093 (SP) 800-38F. December 2012. DOI: [10.6028/NIST.SP.800-38F](https://doi.org/10.6028/NIST.SP.800-38F).
- 2094 [SP800-56A-Rev3] Elaine Barker, Lily Chen, Allen Roginsky, Apostol Vassilev, and Richard Davis.  
2095 *Recommendation for Pair-Wise Key-Establishment Schemes Using Discrete Logarithm Cryp-*  
2096 *tography*. (National Institute of Standards and Technology) NIST Special Publication (SP)  
2097 800-56A Rev. 3. April 2018. DOI: [10.6028/NIST.SP.800-56Ar3](https://doi.org/10.6028/NIST.SP.800-56Ar3).
- 2098 [SP800-56B-Rev2] Elaine Barker, Lily Chen, Allen Roginsky, Apostol Vassilev, Richard Davis, and  
2099 Scott Simon. *Recommendation for Pair-Wise Key-Establishment Using Integer Factorization*  
2100 *Cryptography*. (National Institute of Standards and Technology) NIST Special Publication  
2101 (SP) 800-56B Rev. 2. March 2019. DOI: [10.6028/NIST.SP.800-56Br2](https://doi.org/10.6028/NIST.SP.800-56Br2).
- 2102 [SP800-56C-Rev2] Elaine Barker, Lily Chen, and Richard Davis. *Recommendation for Key-Deriva-*  
2103 *tion Methods in Key-Establishment Schemes*. (National Institute of Standards and Technology)  
2104 NIST Special Publication (SP) 800-56C Rev. 2. August 2020. DOI: [10.6028/NIST.SP.800-56](https://doi.org/10.6028/NIST.SP.800-56)  
2105 [Cr2](#).
- 2106 [SP800-57-P1-R5] Elaine Barker. *Recommendation for Key Management: Part 1 — General*. (Na-  
2107 tional Institute of Standards and Technology) NIST Special Publication (SP) 800-57 Part 1  
2108 Rev. 5. May 2020. DOI: [10.6028/NIST.SP.800-57pt1r5](https://doi.org/10.6028/NIST.SP.800-57pt1r5).

- 2109 [SP800-90A-R1] Elaine Barker and John Kelsey. *Recommendation for Random Number Generation*  
2110 *Using Deterministic Random Bit Generators*. (National Institute of Standards and Technology)  
2111 NIST Special Publication (SP) 800-90A Rev. 1. June 2015. DOI: [10.6028/NIST.SP.800-90Ar1](https://doi.org/10.6028/NIST.SP.800-90Ar1).
- 2112 [SP800-90B] Meltem Sönmez Turan, Elaine Barker, John Kelsey, Kerry McKay, Mary Baish, and  
2113 Michael Boyle. *Recommendation for the Entropy Sources Used for Random Bit Generation*.  
2114 (National Institute of Standards and Technology) NIST Special Publication (SP) 800-90B.  
2115 January 2018. DOI: [10.6028/NIST.SP.800-90B](https://doi.org/10.6028/NIST.SP.800-90B).
- 2116 [SP800-90C-3PD] Elaine Barker, John Kelsey, Kerry McKay, Allen Roginsky, and Meltem Sönmez  
2117 Turan. *Recommendation for Random Bit Generator (RBG) Constructions. Third Public Draft*.  
2118 (National Institute of Standards and Technology) NIST Special Publication (SP) 800-90C 3PD.  
2119 September 2022. DOI: [10.6028/NIST.SP.800-90C.3pd](https://doi.org/10.6028/NIST.SP.800-90C.3pd).
- 2120 [SP800-108-Rev1] Lily Chen. *Recommendation for Key Derivation Using Pseudorandom Func-*  
2121 *tions*. (National Institute of Standards and Technology) NIST Special Publication (SP) 800-  
2122 108 Rev. 1. August 2022. DOI: [10.6028/NIST.SP.800-108r1](https://doi.org/10.6028/NIST.SP.800-108r1).
- 2123 [SP800-135-Rev1] Quynh Dang. *Recommendation for Existing Application-Specific Key Derivation*  
2124 *Functions*. (National Institute of Standards and Technology) NIST Special Publication (SP)  
2125 800-135 Rev. 1. December 2011. DOI: <https://doi.org/10.6028/NIST.SP.800-135r1>.
- 2126 [SP800-185] John Kelsey, Shu-jen Chang, and Ray Perlner. *SHA-3 Derived Functions: cSHAKE,*  
2127 *KMAC, TupleHash and ParallelHash*. (National Institute of Standards and Technology) NIST  
2128 Special Publication (SP) 800-185 (Draft). December 2016. DOI: [10.6028/NIST.SP.800-185](https://doi.org/10.6028/NIST.SP.800-185).
- 2129 [SP800-186-Draft] Lily Chen, Dustin Moody, Andrew Regenscheid, and Karen Randall. *Recommen-*  
2130 *dations for Discrete Logarithm-Based Cryptography: Elliptic Curve Domain Parameters*. (Na-  
2131 tional Institute of Standards and Technology) NIST Special Publication (SP) 800-186 (Draft).  
2132 October 2019. DOI: [10.6028/NIST.SP.800-186-draft](https://doi.org/10.6028/NIST.SP.800-186-draft).
- 2133 [ZkpComRef] ZKProof. *ZKProof Community Reference. Version 0.3*. Published by ZKProof.org.  
2134 Editors: D Benarroch, L Brandão, M Maller, and E Tromer. Contributors since version 0: S Agrawal, T Arcieri, D  
2135 Benarroch, V Bharathan, N Bitansky, S Bowe, L Brandão, B Bünz, R Canetti, A Caro, K Chalkias, J Cinninati, H  
2136 Corrigan-Gibbs, J Daniel, M Dixon, M Dubovitskaya, B Falk, D Genkin, N George, S Goldwasser, A Gupta, J  
2137 Grigg, J Groth, K Gurkan, Y Hang, D Hopwood, Y Ishai, C Jutla, Y Kalai, H Krawczyk, J Law, A Lysyanskaya,  
2138 M Maller, E Morais, Z Manian, A Miller, E Morais, N Narula, R Ostrovsky, G Pacini, O Paneth, R Peralta, A  
2139 Poelstra, T Rabin, M Raykova, A Robinson, R Rothblum, J Rouach, A Scafuro, a shelat, K Solipuram, J Thaler, E  
2140 Tromer, M Varia, M Venkitasubramaniam, M Virza, I Visconti, R Wahby, D Wikstrom, P Wuille, Y Zohar, and A  
2141 Zhang. July 2022. Updated versions at <https://docs.zkproof.org/reference>.