

1

**Draft NISTIR 8214**

2

**Threshold Schemes for  
Cryptographic Primitives**

3

4

*Challenges and Opportunities in Standardization and  
Validation of Threshold Cryptography*

5

6

Luís T. A. N. Brandão

7

Nicky Mouha

8

Apostol Vassilev

9

10

**Draft NISTIR 8214**

11

# **Threshold Schemes for Cryptographic Primitives**

12

*Challenges and Opportunities in Standardization and  
Validation of Threshold Cryptography*

13

14

15

Luís T. A. N. Brandão

16

Nicky Mouha

17

Apostol Vassilev

18

*Computer Security Division*

19

*Information Technology Laboratory*

20

July 2018

21



22

U.S. Department of Commerce

23

*Wilbur L. Ross, Jr., Secretary*

24

National Institute of Standards and Technology

25

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

26  
27

National Institute of Standards and Technology Internal Report 8214  
55 pages (July 2018)

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

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

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

28  
29  
30  
31  
32  
33

**Public comment period: *July 26, 2018 through October 22, 2018***

National Institute of Standards and Technology  
Attn: Computer Security Division, Information Technology Laboratory  
100 Bureau Drive (Mail Stop 8930) Gaithersburg, MD 20899-8930  
Email: [threshold-crypto@nist.gov](mailto:threshold-crypto@nist.gov)

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



72

**Acknowledgments**

73 The authors thank their colleagues who reviewed recent or early versions of this draft  
74 publication. This includes Lily Chen, René Peralta, Ray Perlner, and Andrew Regenscheid.  
75 We look forward to receiving further feedback during the phase of public comments.

76 **Executive Summary**

77 As cryptography becomes ubiquitous, it becomes increasingly relevant to address the  
78 potentially disastrous breakdowns resulting from differences between ideal and real imple-  
79 mentations of cryptographic algorithms. These differences give rise to a range of attacks that  
80 exploit vulnerabilities in order to compromise diverse aspects of real-world implementations.  
81 Threshold schemes have the potential to enable secure modes of operation even when certain  
82 subsets of components are compromised. However, they also present new challenges for  
83 the standardization and validation of security assertions about their implementations.

84 This report is focused on threshold cryptographic schemes, i.e., threshold schemes used  
85 for secure implementations of cryptographic primitives. In an  $f$ -out-of- $n$  threshold scheme,  
86 some security property is tolerant to the compromise of up to  $f$  out of  $n$  components in the  
87 system. The topic is related to traditional “threshold cryptography” (here adopted as an  
88 umbrella term), secure multi-party computation and intrusion-tolerant distributed systems. A  
89 major goal is enhanced protection of secret keys used by implementations of cryptographic  
90 algorithms. More generally, the goal includes the enhancement of a variety of security  
91 properties, such as confidentiality, integrity and/or availability.

92 Secret sharing is a fundamental technique in threshold cryptography. It enables a key (or  
93 some other secret input) to be split into multiple shares distributed across multiple parties.  
94 The “threshold” property translates into the ability to reconstruct the key from a threshold  
95 number of shares, but not from fewer. Thus, splitting a key into shares is an approach for  
96 protecting the secrecy of a key at rest, since the leakage of one or few shares does not reveal  
97 the key. However, this does not solve the problem of how to execute an algorithm that  
98 depends on a key. Particularly, conventional implementations of key-based cryptographic  
99 algorithms require the whole key as input, so if the key had been subject to secret sharing  
100 then the shared key would have to be reconstructed for use by the algorithm.

101 In threshold cryptography, the shares of the key do not need to be recombined to compute  
102 a particular result. Instead, the parties independently or collaboratively calculate shares  
103 of the output, without revealing the input shares to one another. This may be facilitated  
104 by certain mathematical properties, such as homomorphisms, or by cryptographic “secure  
105 computation” protocols. Using the threshold property, the output from the share computation  
106 can then be reconstructed into a final output. This is possible to achieve for NIST-approved  
107 algorithms, such as RSA and DSA signatures, and AES enciphering and deciphering.

108 Threshold schemes can be used, with different security goals, in different applications.  
109 For example: (i) implement a digital signature algorithm without any single component  
110 ever holding the signing key; (ii) implement encryption and decryption correctly even if one  
111 compromised component attempts to corrupt the output; (iii) generate unbiased randomness  
112 even if some randomness contributors are biased or unavailable.

113 The computational paradigm in threshold cryptography brings several security advan-

114 tages but also some potential weaknesses. For example, the use of multiple shares increases  
115 the attack surface to encompass all shares. Thus, the security effect of implementing a  
116 threshold scheme depends on an attack model. It is particularly relevant to consider how  
117 difficult may be the compromise of more than the threshold number  $f$  of components. In  
118 some cases, for example with low  $f$ , the increased attack surface may enable an attack more  
119 efficient and effective than possible against a conventional (non-threshold) primitive.

120 The security effect of a threshold design may also be different across different properties  
121 of interest. For example, while the compromise of one share might not reveal the original key,  
122 the corruption of a single share (or of a computation dependent on it) may affect the integrity  
123 of the output. These observations highlight the need to look at the security benefits brought  
124 by each threshold scheme as a possible tradeoff across properties. In some settings there may  
125 be a strengthening of some security properties while for others the assurance may be reduced.

126 There are techniques designed to mitigate foreseen compromises in more complicated  
127 scenarios. For example, verifiable secret-sharing enables detection of misuse of shares by a  
128 shareholder, thereby enabling operational modes that tolerate this kind of corruption. As an  
129 other example, proactive secret sharing can be used to periodically reshare a secret, thereby  
130 periodically reducing to zero the number of compromised shares. Assuming that old un-  
131 compromised shares are erased, the refreshing makes it more difficult to reach a state where  
132 the number of contemporaneous compromised shares surpasses the compromise threshold.

133 Separating the analysis of different security aspects can sometimes lead to pitfalls. To  
134 avoid such problems it is important to use appropriate formal models of security. At the  
135 same time, it is relevant to assess potential tradeoffs that a threshold cryptographic scheme  
136 induces across different security properties. A system model is also important to charac-  
137 terize different types of attack that a system may be subject to. Specific attacks in the real  
138 world exploit differences between conventional implementations and their idealized versions.  
139 Threshold schemes can be used to improve resistance against some of these specific attacks  
140 that breach specific security properties (e.g., confidentiality of a key) or sets thereof.

141 An abstract security model is not enough to assess the effects of and on a threshold  
142 scheme placed in an adversarial environment. One also needs to characterize implementa-  
143 tion aspects whose variation may affect security. Such characterization helps distinguish,  
144 possibly across different application contexts, the resistance provided against certain classes  
145 of attacks. *To this end, this document proposes that a basis for discussion and comparison of*  
146 *threshold schemes should include the description of several characterizing features. These*  
147 *include the types of threshold, the communication interfaces, the target computing platforms,*  
148 *and the setup and maintenance requirements.*

149 The examples in the document illustrate how security properties can vary depending  
150 on high-level features, on assumed attack vectors and on the type of adversarial goals  
151 and capabilities. On one hand, this helps prevent a possible misconception that a higher  
152 threshold directly means higher security. On the other hand, it also intends to convey that  
153 threshold schemes can be used to implement cryptographic primitives in a more secure

154 way. Altogether, structured security assertions also promote a path for meaningful security  
155 validation of actual implementations.

156 This document considers the benefits of standardizing threshold cryptographic schemes,  
157 possibly along with auxiliary threshold-cryptography primitives. Naturally, there is interest  
158 on threshold schemes for NIST-approved cryptographic primitives. Also of major impor-  
159 tance is the development of corresponding approaches for validation of implementations  
160 of threshold cryptographic schemes. This should be aligned with the current moderniza-  
161 tion process and evolving structure of the testing methodology of the NIST cryptographic  
162 validation programs. Of particular relevance is the development of approaches to enable  
163 automated validation tests with state-of-the-art techniques.

164 The use of well-characterized threshold schemes to implement cryptographic primitives  
165 offers potential security benefits. But what criteria should one use to select from a potential  
166 pool of candidate threshold schemes? What flexibility of features and parameters should a  
167 threshold-cryptographic-scheme standard allow? Should some base primitives be indepen-  
168 dently standardized and/or validated? This document does not offer definitive answers to  
169 these questions. Instead, it motivates the need to develop an objective basis for addressing  
170 them. It also hints at various representative questions to consider, namely about security  
171 assessment, efficiency and applicability, among others.

172 There are important challenges and opportunities related to the standardization of thresh-  
173 old cryptographic schemes. Addressing these may bring about important security improve-  
174 ments to real implementations of cryptographic primitives. Fortunately, there is a plethora  
175 of research work done in the broad area of threshold cryptography, providing useful insights  
176 about possible options, caveats and tradeoffs. Further value can arise from addressing  
177 these challenges with feedback and collaboration from stakeholders, including academic  
178 researchers, industry participants and government representatives.



179 **Table of Contents**

180 **1 Introduction** **1**

181 **2 Fundamentals** **4**

182 2.1 Secret sharing . . . . . 4

183 2.2 Secret resharing . . . . . 5

184 2.3 Threshold cryptography . . . . . 7

185 2.4 Side-channel and fault attacks . . . . . 8

186 2.5 Terminology . . . . . 9

187 **3 Examples** **10**

188 3.1 Threshold signature examples . . . . . 10

189 3.2 Examples of side-channel attacks and countermeasures . . . . . 12

190 **4 Models** **13**

191 4.1 Security considerations . . . . . 14

192 4.2 Types of attack . . . . . 17

193 4.3 System model . . . . . 19

194 **5 Characterizing features** **22**

195 5.1 Threshold values . . . . . 23

196 5.2 Communication interfaces . . . . . 25

197 5.3 Target computing platforms . . . . . 26

198 5.4 Setup and maintenance . . . . . 30

199 **6 Validation of implementations** **32**

200 6.1 The existing CMVP and FIPS 140-2 . . . . . 32

201 6.2 Integration of threshold cryptographic schemes . . . . . 33

202 **7 Criteria for standardization** **34**

203 **8 Conclusions** **36**

204 **References** **38**

205 **List of Figures**

206 1 Illustration of Blakley secret sharing . . . . . 5

207 2 Illustration of share randomization in Blakley secret sharing . . . . . 6

208 **List of Tables**

209 1 Representative attack types . . . . . 17

210 2 Characterizing features of threshold schemes . . . . . 23

## 211 **1 Introduction**

212 Protecting sensitive information from unauthorized disclosure has always been challenging.  
213 “Two may keep counsel, putting one away,” William Shakespeare wrote in “Romeo and Juliet”  
214 (1597) [Sha97]. Later, in “Poor Richard’s Almanack — 1735” [Sau34], Benjamin Franklin  
215 observed that “Three may keep a secret, if two of them are dead.” Today, cryptography is a  
216 primary means of protecting digital information. In modern cryptography the algorithms  
217 are well known but the keys are secret. Thus, the effectiveness of encrypting data hinges on  
218 maintaining the secrecy of cryptographic keys. However, this is difficult in conventional  
219 implementations, as keys are usually stored in one place on a device, and used there to  
220 run the algorithm. Devices, much like people, are not completely dependable guardians of  
221 secrets. Does this mean that keys are the *Achilles’ heel* of cryptography?<sup>1</sup>

222 The localization of a key, for use by an algorithm, is susceptible to enabling leaking  
223 it out. For example, the internal state of a conventional implementation might be compro-  
224 mised through a bug such as Heartbleed [DLK<sup>+</sup>14, NVD14], Spectre [KGG<sup>+</sup>18, NVD18a,  
225 NVD18b] and Meltdown [LSG<sup>+</sup>18, NVD18c], letting an attacker read private memory  
226 locations, including secret keys contained therein. Another example is the cold-boot at-  
227 tack [HSH<sup>+</sup>09], which allows recovery of keys from the dynamic random access memory  
228 (DRAM) of a computer, even seconds to minutes after it has been removed from the device.  
229 Some attacks inject faults into the computation, for example by changing the supply voltage.  
230 An example is the “Bellcore” attack [BDL97, ABF<sup>+</sup>03], where a fault induces an incorrect  
231 computation whose output reveals a secret key. Other attacks obtain information through  
232 a side channel, such as the execution time, the amount of energy it consumes, or the elec-  
233 tromagnetic emanations it produces. Many of these fall into the category of non-invasive  
234 attacks, which can be performed without direct physical contact with components within  
235 the device. Attacks that exploit leakage of key-dependent information can lead to disastrous  
236 scenarios in which the master key used to encrypt and authenticate device firmware becomes  
237 compromised [RSWO17].

238 To counter the inherent security risks of handling secret keys in conventional implemen-  
239 tations of cryptographic algorithms, technical approaches have emerged that split the secret  
240 key into two or more shares across different components or parties. For example, upon using  
241 secret-sharing the compromise of one (or more, but not all) of the shares does not reveal  
242 information about the original key. Using appropriate threshold techniques, the shares can  
243 then be separately processed, leading the computation to a correct result as if the original  
244 secret key had been processed by a classic algorithm. The threshold approach can thus  
245 significantly increase the confidentiality of secret keys in cryptographic implementations.

246 In this report, we focus on threshold schemes applied to cryptographic primitives. In  
247 an  $f$ -out-of- $n$  threshold scheme, some security property is tolerant to the compromise of up  
248 to  $f$  out of  $n$  components in the system. This paradigm brings several security advantages

---

<sup>1</sup>Some portions of writing were adapted from text appearing at a previous short magazine article [VMB18].

249 but also some potential weaknesses. For example, the use of multiple shares increases the  
250 attack surface to encompass all shares. Thus, the security effect of implementing a threshold  
251 scheme depends on an attack model. It is particularly relevant to consider how difficult may  
252 be the compromise of more than the threshold number  $f$  of components. In some cases, for  
253 example with low  $f$ , the increased attack surface may enable an attack more efficient and  
254 effective than possible against a conventional (non-threshold) primitive.

255 The threshold concept can apply to security properties of interest beyond the secrecy of  
256 keys. For example, it is useful to enable availability and integrity of computations in spite of  
257 malfunctioning of some of its components. Traditional techniques of fault tolerance often  
258 achieve such resistance when considering random or predictably modeled faults. However,  
259 we are specially interested in resistance against targeted attacks, which can be malicious and  
260 arbitrary. Considering a wide scope of security goals, threshold schemes can exist in several  
261 flavors, depending on the security aspects they address and the techniques used. There are  
262 challenges in ensuring the simultaneous upholding of diverse security properties, such as  
263 secrecy of key material, correctness of outputs and continued availability.

264 In fact, the security impact of a threshold design may be different across different proper-  
265 ties of interest. For example, in some schemes the compromise of one share might not reveal  
266 the original key but the corruption of a single share (or of a computation dependent on it)  
267 may affect the integrity of the output. These observations highlight the need to look at the  
268 security benefits brought by threshold cryptography as a possible tradeoff across properties.

269 The basic security model for cryptographic algorithms assumes an ideal black box, in  
270 which the cryptographic computations are correct and the internal states are kept secret.  
271 For example, such ideal constructs have no side channels that could leak secret keys. This  
272 model contrasts with the reality of conventional implementations, which can be subject to  
273 attacks that exploit differences between the ideal and real worlds. Threshold schemes deal  
274 with some of those differences, by providing tolerance against the compromise of several  
275 components. They may also hinder the exploitation of existing compromises (such as noisy  
276 leakage) from a set of components, e.g., providing resistance against side-channel attacks.

277 A separate analysis of different security properties may lead to some pitfalls. Some  
278 formal models of security are useful to avoid them. The ideal-real simulation paradigm,  
279 common to analysis of secure multi-party computation protocols, combines the notion of  
280 security into a definition of an ideal world. This abstraction captures an intended application  
281 in an ideal world, then allowing security properties to be derived therefrom. Complementary,  
282 a system model is also important to characterize different types of attack that a system may  
283 be subject to. Specific attacks in the real world exploit differences between conventional  
284 implementations and their idealized versions. Some of these may target breaching specific  
285 security properties (e.g., confidentiality of a key) or sets thereof. There is a particular interest  
286 in understanding how threshold schemes can be used to improve resistance against these  
287 specific attacks. It is also relevant to assess potential tradeoffs that a threshold cryptographic  
288 scheme induces across different security properties.

289 There are techniques designed to mitigate foreseen compromises in more complicated  
290 scenarios. For example, verifiable secret-sharing enables detection of misuse of shares by a  
291 shareholder, thereby enabling operational modes that tolerate this kind of corruption. As an-  
292 other example, proactive secret sharing can be used to periodically reshare a secret, thereby  
293 periodically reducing to zero the number of compromised shares. However, an abstract  
294 security model is not enough to assess the effects of and on a threshold scheme placed in  
295 an adversarial environment. One also needs to characterize implementation aspects whose  
296 variation may affect security. These include the types of threshold, the communication  
297 interfaces, the target computing platforms, and the setup and maintenance requirements.

298 Altogether, the security assertions made with respect to an instantiated set of features  
299 provide a path for security validation of actual implementations. Of particular interest are  
300 approaches that enable automated validation tests with state-of-the-art techniques. The  
301 use of well-characterized threshold cryptographic schemes to implement cryptographic  
302 primitives offers potential security benefits. It is thus important to develop objective criteria  
303 for selecting from a potential pool of candidate threshold schemes.

304 **Audience.** This document is targeted, with varying goals, at a diverse audience. Internally  
305 for NIST, the goal is to initiate a discussion about threshold schemes for cryptographic prim-  
306 itives. This motivated the inclusion of representative questions relevant to standardization.

307 The document is also written for people with managerial/policy responsibilities in devel-  
308 opment and/or adoption of cryptographic services and modules. For such an audience, the  
309 document highlights critical aspects of the security of implementations that can be signifi-  
310 cantly affected by nuances in the system model and the employed threshold techniques. Sev-  
311 eral simple examples are provided, including some based on classic secret sharing schemes.

312 The text is also directed to experts in cryptography from academia and industry. For  
313 them, the document is an invitation to engage with NIST in a collaborative effort to resolve  
314 the open questions related to the standardization of threshold schemes for cryptographic  
315 primitives and the corresponding guidelines for implementation validation.

316 It is useful to further clarify one intentional design aspect related to the references to re-  
317 lated work. This document intends to initiate a discussion that may lead NIST to standardize  
318 threshold schemes for cryptographic primitives. For that purpose, we sought to convey in  
319 a balanced way that there are feasible threshold approaches, but without showing particular  
320 preferences. In fact, we specifically opted to avoid an assessment of the most recent works,  
321 preferring instead to exemplify precursory threshold techniques. Therefore, we do not make  
322 an exhaustive analysis and do not try to include the depth and nuances typical of a research  
323 paper or a technical survey. We hope that a thorough assessment of state-of-the-art threshold  
324 approaches can be subsequently performed with an inclusive participation of stakeholders.

325 **2 Fundamentals**326 **2.1 Secret sharing**

327 Secret sharing is based on splitting the key into multiple shares. For example, to split key  
 328  $K$  into three shares  $K_1$ ,  $K_2$ , and  $K_3$ , we randomly select shares  $K_1$  and  $K_2$  from the same key  
 329 space as  $K$ , and let the third share  $K_3 = K_1 \oplus K_2 \oplus K$  be the one-time pad encryption of  $K$ ,  
 330 where  $\oplus$  is the exclusive OR operation if the keys are bit-strings. No two shares provide  
 331 any information about the secret key — all shares are required to recover  $K$ . The described  
 332 scheme has a “3-out-of-3” property. More generally,  $k$ -out-of- $n$  secret-sharing schemes can  
 333 be defined, for any integers  $n$  and  $k$  satisfying  $n \geq k \geq 1$ . Such secret-sharing schemes were  
 334 independently developed in 1979 by Shamir [Sha79] and Blakley [Bla79]. There, any  $k$   
 335 parties together can recover a secret shared across  $n$  parties, but  $k - 1$  parties together do  
 336 not know anything about the secret.

337 With the help of Fig. 1, we describe an example of Blakley’s scheme for  $k = 2$  and  $n = 3$ ,  
 338 with some simplifications for illustration purposes. The secret is the  $x$ -coordinate ( $x_s$ ) of the  
 339 point  $P(x, y)$  in the two-dimensional plane (see Fig. 1(a)). A non-vertical line in the plane is  
 340 defined as a set of points  $(x, y)$  satisfying  $y = hx + g$  for some constants  $h$  and  $g$ . If Alice  
 341 obtains coefficients  $h_A$  and  $g_A$  for some line  $\{(x, y) : y = h_Ax + g_A\}$ , containing the point  $P$ ,  
 342 this does not give Alice any advantage in discovering its  $x$ -coordinate  $x_s$  (see Fig. 1(b)). This  
 343 is because the definition of the line does not provide any special information about any point  
 344 in the line, i.e. all points in the line (and all  $x$ -coordinates) are equally likely. In practice,  
 345 lines are selected only from a finite space of lines, e.g., with all coefficients being integers  
 346 modulo some prime number  $Q$ , and the lines themselves are finite collections of points, e.g.,  
 347 with  $x$  and  $y$  being also integers modulo  $Q$ .

348 Similarly, if Bob and Charlie obtain coefficients of other lines that pass through the  
 349 same point  $P$ , individually they cannot determine  $P$ . However, any two together — Alice  
 350 with Bob, or Alice with Charlie, or Bob with Charlie — can easily compute  $P$  as the  
 351 intersection of their lines (see Fig. 1(c)). We have thus described a 2-out-of-3 secret-  
 352 sharing scheme. To build a  $k$ -out-of- $n$  Blakley scheme for some  $k > 2$ , one considers  
 353 hyperplanes  $y = h_1x_1 + \dots + h_{k-1}x_{k-1} + g$  that intersect in a single point  $P(x_1, \dots, x_{k-1}, y)$   
 354 in the  $k$ -dimensional space. The coefficients  $h_i$  are non-zero and  $g$  is an arbitrary constant.  
 355 Choosing  $n \geq k$  such hyperplanes, one can distribute the corresponding coefficients to  $n$   
 356 different parties. Then any  $k$  parties together can compute efficiently the intersection point  $P$ .  
 357 The prime modulus  $Q$  must be larger than the secret  $x_s$  and larger than the number  $n$  of parties.

358 Shamir secret sharing is based on the observation that any set of  $k$  distinct points  
 359 determines completely a polynomial of degree  $k - 1$ . For example, consider a set of positive  
 360 integer coefficients  $c_0, c_1, \dots, c_{k-1}$  and define the polynomial  $f(x) = c_0 + c_1x + \dots + c_{k-1}x^{k-1}$ .  
 361 Typically, the secret is the coefficient  $c_0 = f(0)$  and each party  $i$  receives as share the point

<sup>5</sup>The humanoid cliparts are from [clker.com/clipart-\\*.html](http://clker.com/clipart-*.html), where \* is 2478, 2482 and 2479.

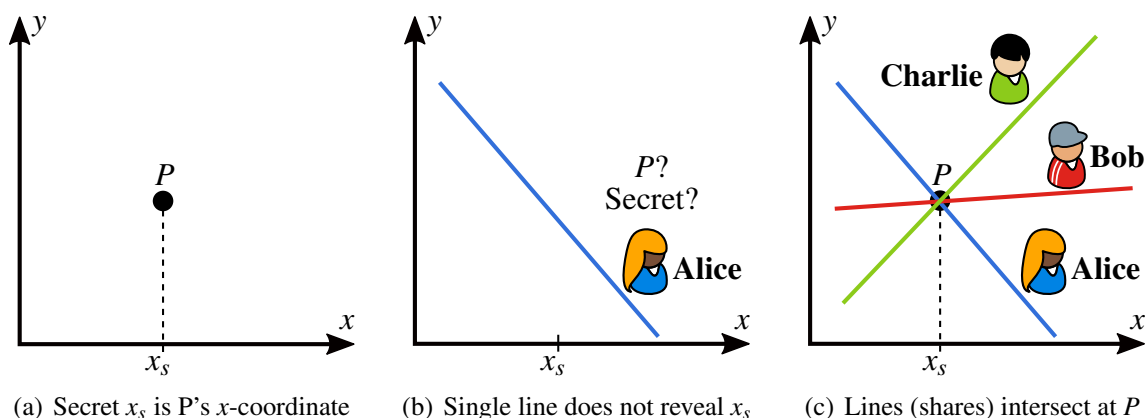


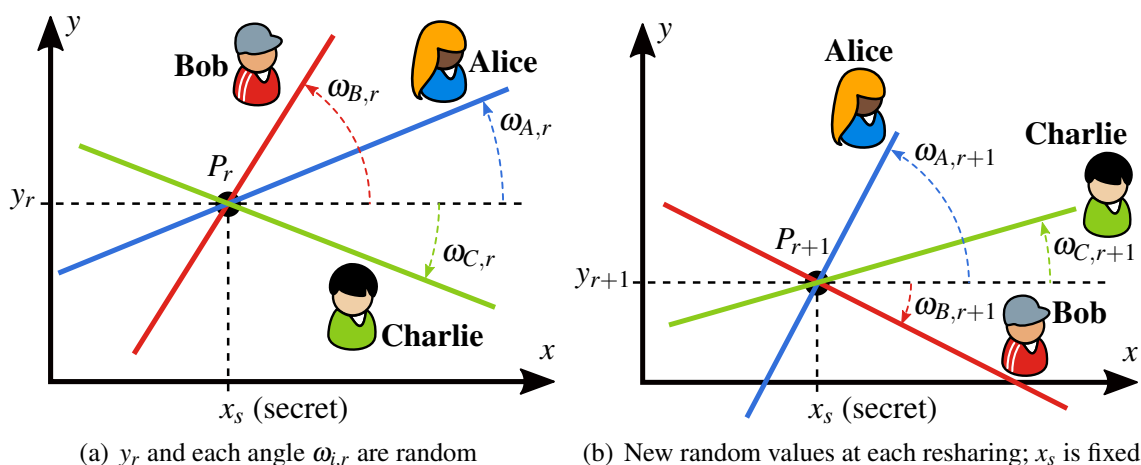
Figure 1. Illustration<sup>5</sup> of Blakley secret sharing

362  $(i, f(i))$ , where  $i$  is a positive integer distinct for each party (e.g.,  $1, 2, \dots, n$ ). Then, any  
 363 set of  $k$  parties can reconstruct  $f(x)$ , and therefore compute the secret  $f(0)$ , whereas  $k - 1$   
 364 parties cannot. All coefficients are based on finite field arithmetic defined in terms of a  
 365 prime number  $Q$ . Since each party must receive a distinct point, and that point must not  
 366 be  $(0, f(0))$ , the modulus  $Q$  must be larger than the number  $n$  of parties. The points on the  
 367 curve are thus defined as  $(x, f(x) \bmod Q)$  and the secret and any other coefficient are integers  
 368 between  $0$  and  $Q - 1$ . This ensures that no information from the secret can be recovered  
 369 from incomplete sets of (i.e., with less than  $k$ ) points on the curve.

370 Shamir and Blakley's schemes are information-theoretic secure, which means that indeed  
 371 there is no information about the key in a standalone set of  $k - 1$  shares. This means that  
 372 the scheme can in practice be used to share very small secrets (e.g., only a few bits),  
 373 independently of the application. If, however, the sharing is applied to a cryptographic key  
 374 required to be larger than some security parameter, e.g., 256 bits, then the corresponding  
 375 prime  $Q$  must be correspondingly large. Alternatively, the secret sharing could be applied in  
 376 parallel to independently share portions of a secret. While information-theoretic security  
 377 may be an advantage, the property requires that each share is of the same size as the secret,  
 378 thus meaning that the overall size of all shares is  $n$  times the size of the secret. In contrast,  
 379 there are secret-sharing schemes with reduced optimal size, at the cost of guaranteeing only  
 380 computational (i.e., cryptographic) security [Kra94]. There, the size of each share can be up  
 381 to  $k$  times smaller than the size of the secret — this is specially useful if secret sharing is to  
 382 be used to share large amounts of data.

## 383 2.2 Secret resharing

384 The need to compute new random shares for the same original secret key often arises in  
 385 practice. It may happen that over time some ( $< k$ ) shares are compromised [OY91], thus  
 386 creating a need to compute new shares and discard the old ones. Resharing can even be



**Figure 2. Illustration of share randomization in Blakley secret sharing**

387 proactive [HJKY95], e.g., at regular intervals in time and not as a direct response to a  
 388 detected compromise.

389 **Resharing in Blakley’s scheme.** We continue here the 2-out-of-3 example of Blakley’s  
 390 scheme, where two parties are required to reconstruct a secret  $x_s$  shared among three parties.  
 391 Each resharing of  $x_s$  requires re-randomizing the point  $P$  along the vertical line that defines  
 392 the secret. In other words, for each randomization iteration  $r$  a random  $y$ -coordinate  $y_r$   
 393 is sampled, defining a new point  $P_r = (x_s, y_r)$ . Then, the new share (a line) for each party is  
 394 also randomized, subject to the constraints that all new lines intersect at the new point  $P_r$   
 395 and are different from one another. With this construction, a single party (Alice, Bob, or  
 396 Charlie) still cannot gain any useful insight into the reshared secret  $x_s$ . This is because at  
 397 each new resharing  $r$  the point  $P_r$  where the three lines intersect is chosen randomly in the  
 398 vertical line that passes through the secret.

399 For visual intuition, we illustrate in Fig. 2 a parametrization based on angles. A line  
 400 through a point  $P$  in the plane can be parametrized in terms of its angle  $\omega$ , in the interval  
 401  $(-\pi/2, \pi/2]$ , with respect to the  $x$  axis. Thus, for each resharing iteration  $r$  we attribute to  
 402 each party  $i$  a new random angle  $\omega_{i,r}$ . An angle is not sufficient to define a line, so some other  
 403 reference point is required. The reference cannot be point  $P_r$ , since that would reveal the se-  
 404 cret, but could for example be the  $x$ -coordinate where the line intersects with the  $x$ -axis. How-  
 405 ever, this is not even a concern because in practice the parametrization used is not based on  
 406 angles, but rather on polynomial coefficients. In other words, the share (a line) is not revealed  
 407 as  $(P, \omega)$  but rather as  $(g, h)$ , where  $y = hx + g$  is the equation that defines the same line.

408 For each new iteration  $r + 1$ , one computes a new point  $P_{r+1} = (x_s, y_{r+1})$  and new random  
 409 lines for each party. These lines, passing through point  $P_{r+1}$  correspond to new random  
 410 angles, as illustrated in Fig. 2(b). The dealer (i.e., the party selecting new shares) must



411 ensure that the lines of different parties to not overlap, i.e., that they do not have the same  
412 angles. Concretely, this means that  $\omega_{i,r} \neq \omega_{j,r}$  for  $i, j \in \{A, B, C\}$  and  $i \neq j$ .

413 **Resharing in Shamir’s scheme.** Share re-randomization can also be done with Shamir  
414 secret sharing. There, the fixed secret is  $c_0 \bmod Q = f(0) \bmod Q$ . At each random-  
415 ization iteration  $r$ , one chooses random coefficients  $c_{1,r}, \dots, c_{k-1,r}$  for a new polynomial  
416  $f_r(x) = c_0 + c_{1,r}x + \dots + c_{k-1,r}x^{k-1}$  satisfying  $f_r(0) = c_0$ . The new shares are then points  
417 evaluated with  $f_r$ . Concretely, each party  $i$ , for  $i = 1, 2, 3, \dots$  receives  $f_r(i)$  as its new share.

418 **Note:** several elements of secret-sharing are standardized by ISO/IEC [ISO16, ISO17].

### 419 2.3 Threshold cryptography

420 We take broad input from several research areas with traditionally distinctive names, but  
421 with a strong relation to threshold schemes. Since we are focused on the implementation  
422 of cryptographic primitives, we adopt the umbrella term “threshold cryptography” to de-  
423 note our area of interest. The expression “threshold cryptography” has been traditionally  
424 used to refer to schemes where some computation is performed over secret shares of in-  
425 puts [DF90, DSDFY94]. Usually, the setting is such that the shares are used to compute  
426 something useful, but without being revealed across parties. Often, a main security goal is  
427 secrecy of cryptographic keys, but a variety of other security properties, such as integrity  
428 and availability, may also be a motivating drive. Achieving these properties is possible  
429 based on a variety of techniques. For example, integrity may in some settings be enhanced  
430 based on verifiable secret sharing schemes [AMGC85, Fel87] and/or zero-knowledge proofs  
431 [GMR85, BFM88], allowing checking whether shares are used consistently. Specifically, a  
432 threshold scheme can be made robust against adversarially induced inconsistencies in shares  
433 or in related computations, outputting correct results in spite of up to a threshold number  
434 of compromised parties [GRJK00]. While we focus on secure implementations of cryp-  
435 tographic primitives, the actual threshold techniques may also include non-cryptographic  
436 techniques, e.g., simple replication and majority voting.

437 One main area of related research is “secure multi-party computation” (SMPC) [Yao86,  
438 GMW87]. It allows mutually distrustful parties to compute functions (and randomized  
439 functionalities) of their combined inputs, without revealing the corresponding inputs to  
440 one another. This can be useful for threshold schemes even if the inputs of different  
441 parties are not shares of some key and/or if the actual computation requires interaction  
442 between parties. In usual SMPC descriptions, the parties themselves are stake-holders  
443 of the secrecy of their input, e.g., in the millionaire’s problem [Yao82] and in secure set  
444 intersection [FNP04]. Conversely, threshold schemes are often envisioned at a higher level,  
445 where the threshold entity has neutral interest for the outcome, and is in fact just a service  
446 provider (of cryptographic services) to a set of users/clients. In other words, threshold



447 schemes do not encompass all that exists in the realm of SMPC, and vice-versa there are  
448 threshold schemes not based on SMPC.

449 Threshold schemes can also be based on elements from the “distributed systems” re-  
450 search area, where fault and intrusion tolerance are main topics. Common properties of  
451 interest in distributed systems are liveness (making progress even in the face of concurrent  
452 execution/requests) and safety (ensuring consistency of state across multiple parties). Why  
453 would this be relevant for threshold cryptography? As an example, consider implementing  
454 a multi-party threshold version of a full-fledged cryptographic platform. Such platform  
455 would perform a variety of cryptographic operations, some using secret keys, and based  
456 on requests by users whose credentials and authorization profiles may be updated across  
457 time. Now we could ask: in a setting where *availability* (of cryptographic operations) is  
458 a critical property, and where the system is supposed to operate even in cases of network  
459 *partition* (i.e., even if some parties in the threshold scheme cannot inter-communicate), can  
460 *consistency* (of state, e.g., credentials, across different parties) be simultaneously satisfied  
461 under concurrent executions? This is a kind of “distributed systems” problem relevant for  
462 threshold schemes. There are settings [Bre12] where these three properties (consistency,  
463 availability and partition tolerance) cannot be guaranteed to be achieved simultaneously.

#### 464 **2.4 Side-channel and fault attacks**

465 The secrecy of keys can be compromised by the leakage of key-dependent information  
466 during computations. This is possible even without direct physical contact with components  
467 within the device. For example, the time taken, the power consumed, and the electromagnetic  
468 radiation emanated by a device can be measured without penetrating the device enclosure.

469 We will assume that, regardless of whether the computation is in hardware or software,  
470 the device that performs the computation consists of some circuit with wires connecting to  
471 logical gates and memory cells. Then, the attacker’s view of the circuit elements may be  
472 noisy (the *noisy leakage* model [CJRR99]), or the attacker may be limited by the number  
473 of wires of the circuit that it can observe within a certain period of time (the *probing*  
474 model [ISW03]). The noisy leakage model and probing model have been unified [DDF14].  
475 In both models, under some reasonable assumptions on the statistical distributions of side-  
476 channel information, the complexity of a side-channel attack of a suitable implementation  
477 with an  $n$ -out-of- $n$  secret-sharing increases exponentially with the number of shares.

478 As such, side channel attacks on secret-shared implementations become infeasible if the  
479 number of shares is sufficiently high, and is further thwarted when the shares are refreshed  
480 before the attacker can collect enough side-channel information. Further refinements of  
481 the model take transient behavior (“glitches”) of the transistors into account, which can  
482 be handled by Threshold Implementations (TI) [NRR06] or by “lazy engineering” to just  
483 increase the number of shares [BGG<sup>+</sup>14].

484 Besides the aforementioned side-channel attacks, an attacker may also obtain key-  
485 dependent information by injecting a fault into the computation, and then observing the  
486 outputs [BDL97]. To inject the fault, the attacker may, for example, apply a strong external  
487 electromagnetic field. Note that the injection of faults may also introduce errors in the outputs  
488 of the computation, thereby violating the integrity of the outputs. If the threshold scheme  
489 is endowed with the ability to detect which shares have errors, and if the threshold scheme  
490 does not require all shares to be present, it can resist temporary and permanent faults in parts  
491 of the computation. This would provide resistance against a wide range of fault attacks.

## 492 2.5 Terminology

493 We borrow terminology from different research areas, with some overlap, using several  
494 terms that share similar connotations. Sometimes (but not always) they are interchangeable  
495 in the context of  $f$ -out-of- $n$  threshold schemes, where  $f$  denotes a threshold number of  
496 components that can be compromised without violating some security property of interest  
497 in the overall system. Some informal correspondences:

- 498 • **Active/byzantine/malicious**: characterization of compromised nodes, or of an adversary,  
499 when being able to arbitrarily deviate or induce deviations from a protocol specification.
- 500 • **Agent/component/node/party/share**: a constituent part of an implemented threshold  
501 scheme, affecting the prosecution of a functional goal (a cryptographic operation, in our  
502 context) to be achieved by a collective of parts; most often used to denote one of the  $n$   
503 parts whose compromise counts towards the threshold  $f$ ; when the context is clear, some  
504 terms can designate parts outside of the threshold composition.
- 505 • **Aggregator/broker/combiner/dealer/proxy/relay**: an agent with a special role in aiding  
506 the setup, execution and/or maintenance of a threshold protocol; usually not accounted in  
507  $n$ , except if explicitly stated as such (e.g., the case of a primary node).
- 508 • **Bad/compromised/corrupted/controlled/faulty/intruded**: state of a node, whereby it  
509 departs from an ideally healthy state, and starts being counted towards the threshold  $f$ .
- 510 • **Client/user**: an agent, not in the threshold set of components, who is a stake-holder of  
511 the result of a cryptographic computation, typically the requester for that computation.
- 512 • **Compromise/corruption/intrusion**: a process by which a node transitions from an  
513 ideally healthy state to a compromised state and/or by which it remains therein.
- 514 • **Good/healthy/honest/recovered**: ideal state of a node, not yet compromised by an  
515 adversary, but susceptible to attacks.
- 516 • **Honest-but-curious/Leaky/Passive/Semi-honest**: characterization of compromised com-  
517 ponents, or of an adversary, when the internal state of the former is exfiltrated by the later,  
518 but without altering the computations and message-exchanges specified by the protocol.

519 • **Recovery/refresh/rejuvenation/replacement:** transitioning of a node or nodes from a  
 520 (possibly) bad state back to a good state; nuances include update, reversion, change and  
 521 reset of internal states, as well as effective replacement of physical components.

522 The above notes simply intend to convey intuition helpful for reading the document. We  
 523 do not undertake here the goal of unifying terminology from different areas. Cited references  
 524 in the text provide necessary context. The encyclopedia of cryptography and security [TJ11]  
 525 and the NIST glossary of security terms [Kis13] provide additional suggestions.

## 526 3 Examples

### 527 3.1 Threshold signature examples

528 **Basic threshold computation on secret shares.** Now let us proceed to construct a thresh-  
 529 old scheme for digital signatures. First, we recall the RSA (Rivest-Shamir-Adleman)  
 530 signature scheme [RSA78], which defines the public key as  $(N, e)$  and the private key  
 531 as  $d$ , such that  $ed = 1 \pmod{\phi(N)}$ . Here, the modulus  $N$  is a product of two large secret  
 532 primes and  $\phi$  is Euler's totient function. Then, the RSA signature for a (possibly hashed)  
 533 message  $m$  is defined as  $s = m^d \pmod{N}$ . Anyone possessing the public key can verify  
 534 the signature by checking  $s^e = m^{ed} = m \pmod{N}$ . To obtain a threshold variant of this  
 535 signature scheme, we split the private key  $d$  into three shares  $d_1, d_2,$  and  $d_3$ , such that  
 536  $d_1 + d_2 + d_3 = d \pmod{\phi(N)}$ . Now, without reconstructing  $d$ , it is possible to first process  
 537 the message independently using each of the shares:  $s_1 = m^{d_1}, s_2 = m^{d_2},$  and  $s_3 = m^{d_3}$ ; and  
 538 then compute the signature  $s = s_1 s_2 s_3$ . Note that this is indeed a valid RSA signature, as  
 539  $s_1 s_2 s_3 = m^{d_1 + d_2 + d_3} = m^d \pmod{N}$ . This simple threshold RSA signature scheme mitigates  
 540 the risk of exposing the potentially high-value private key  $d$ , which doesn't appear in any of  
 541 the three shares that are used in the actual computations. Thus, compromising any one of  
 542 the shares, and even two of them, poses no threat of exposing  $d$ . Moreover, frequent updates  
 543 to the key shares ( $d_1, d_2,$  and  $d_3$ ) would reduce the window of opportunity for attacks and  
 544 thereby further reduce the risk. Refreshing can even occur after every signature.

545 **A  $k$ -out-of- $n$  threshold scheme.** In the above example, all shares must be present. This  
 546 might be impractical in situations where one or more of the shares become unavailable. For  
 547 such cases, a  $k$ -out-of- $n$  threshold scheme could be used when at least  $k$  shares are available.  
 548 For RSA signatures, one can use a 2-out-of-3 secret-sharing scheme, and a corresponding  
 549 threshold variant of RSA [Sho00]. Then, in the case of one share being irrecoverably lost  
 550 or breached, the private signature key  $d$  remains intact, available, and not breached. This  
 551 means that one can continue to use the same public key to verify the signature's correctness.

552 In contrast, when a conventional implementation is breached, the corresponding pub-  
 553 lic/private key pair would have to be revoked and a new pair issued. Typically this also

554 requires an external certification of the public key by a certificate authority and propagating  
555 it to all relying parties. In addition, a 2-out-of-3 threshold signature scheme becomes more  
556 resilient to future share losses if it continuously refreshes the key shares, provided that at  
557 most one is compromised at any given time. Note that in a scheme composed of three sepa-  
558 rate conventional RSA implementations with independent keys, refreshing would require  
559 updating the public/private key pairs, along with all entailing inconveniences.

560 **Avoiding the dealer.** In the above descriptions, an implicit trusted party knows the secret  
561 and performs each secret-sharing operation. Particularly, the threshold RSA examples based  
562 on a common modulus  $N$  required the dealer to know the secret prime factorization. Without  
563 knowledge of such factorization, it is not currently known how to correctly select such  
564 modulus and prove its correctness. Thus, the selection of secrets does not lend itself to a  
565 straightforward efficient threshold computation. Nonetheless, such threshold selection, even  
566 for RSA keys, can still be done based on SMPC protocols [BF97]. By using zero-knowledge  
567 proofs [vdGP88] the needed property on  $N$  can also be proven without revealing the secret.

568 Schemes based on different assumptions can enable a more straightforward selection  
569 and verification of the validity of public elements. For example, this is possible based on  
570 assumptions of intractability of computing discrete logarithms in certain groups of known  
571 order. If the group parameters can be verified as correct in a standalone procedure, then no  
572 one requires knowing any secret knowledge about the group. Furthermore, if the selection is  
573 made in a way that avoids the possibility of a trapdoor being known, then the parameters can  
574 be trusted by anyone. The intractability assumption can then, for fixed security parameters,  
575 be accepted for universal parameters of a group (e.g., [Ber06]). In particular, this can  
576 facilitate a respective threshold mechanism, so that a secret key never exists locally at any  
577 entity. For example, one can then define a dealer-absent threshold version of a public key  
578 generation (the result of an exponentiation), such that each party knows one share of the  
579 secret key (a discrete logarithm) [Ped91].

580 **Other constructions.** The above examples focused on threshold schemes where the secret-  
581 key is shared, and then a threshold scheme enables a generation of a signature identical to the  
582 non-threshold manner. A feature of those schemes is that the final signature is identical to a  
583 non-threshold one, thereby being inherently efficient in size (i.e., not incurring an increase  
584 with the threshold parameter). Such schemes also have the property that the identities of  
585 the signatories remain secret to the external users interested in verifying the correctness  
586 of a signature. However, some settings may favor the identifiability of signatories, e.g.,  
587 as an accountability and credibility feature. Each signatory might also prefer retaining an  
588 individual public credential, not wanting to use a private-key share associated with a common  
589 public key. Even in this setting it is possible to devise short threshold signatures, with size  
590 equal to a non-threshold signature. Concretely, “multi-signature” schemes [IN83, MOR01]  
591 enable multiple parties, with independent secret-public key pairs, to jointly produce a

592 common short signature.<sup>6</sup>

593 A multi-signature scheme can be used as a threshold signature scheme where the  
594 application layer, and possibly the user, has added *flexibility* to define which subsets of  
595 signatories determine a valid signature, i.e., beyond structures defined by a pre-determined  
596 threshold number. For example, a multi-signature may be defined as valid if it contains one  
597 signature from each of three groups of individuals in different roles in an organization. The  
598 verification procedure then depends on the set of independent public keys. For example,  
599 these schemes can be easily based on Schnorr signatures [Sch90, BN06].

600 To complement the resilience in the face of compromise, signatures can also be im-  
601 plemented with a “forward security” property [And02]. Such schemes can be based on  
602 an evolving private key, while the public key remains fixed, so that even a future key  
603 leakage will not allow the adversary to forge past messages, assuming the signer erases  
604 past keys [BM99]. To some extent, this property has some conceptual similarity to the  
605 refreshing we previously described in the RSA example. This property can be achieved also  
606 for threshold signatures [AMN01], including the case of multi-signatures [SA09].

607 In summary, we showed by examples that “threshold signature schemes” can be based  
608 on secret-shared computation of regular signatures or on multi-signatures, with or without a  
609 dealer, with or without robustness, and possibly with forward security.

610 Several of the exemplified threshold schemes take advantage of group homomorphic  
611 properties. While such properties are not applicable in every cryptographic primitive,  
612 threshold computation can still in general be obtained via secure multi-party computation.

### 613 3.2 Examples of side-channel attacks and countermeasures

614 Timing attacks were first presented by Kocher [Koc96], and have been shown to be easy to  
615 perform on a variety of cryptographic algorithms. An advantage of timing attacks is that  
616 no specialized equipment is required. Because they do not require physical access to the  
617 system, they may even be performed remotely over the Internet [BB03].

618 A possible countermeasure against timing attacks is to ensure that the implementation is  
619 “constant time,” that is, that its execution time does not depend on the value of the secret key.  
620 This turns out to be surprisingly difficult for many commonly-used implementations. The  
621 reason is that it may not be sufficient to have “constant-time” source code, that is, source  
622 code without key-dependent branches or memory accesses [Ber05].

623 Even worse, an implementation that is free of timing attacks on one platform, may  
624 be vulnerable on another platform. This can happen, for example, when source code that

---

<sup>6</sup>These should not be confused with “group signatures” [CvH91], where a member of a group signs a message, while proving group membership but remaining anonymous with respect to its identity within the group.

625 contains multiplication operations is compiled with a different runtime library [KPVV16],  
626 or when the same binary is executed on a different processor [Por18].

627 The execution time of the program, however, is just one example of a side channel.  
628 Implementations in hardware and software may also leak through other side channels, such  
629 as power consumption or electromagnetic radiation. The limitation of the currently-known  
630 countermeasures (such as “constant-time” implementations, dual-rail logic, or electromag-  
631 netic shielding) is that they usually do not get rid of all the leakage, but may still be  
632 vulnerable to higher-order or data-dependent leakages.

633 To protect against side-channel attacks, the framework of threshold cryptography can  
634 provide a promising starting point. If the implementation is split into a number of “parties,”  
635 such that no single party holds the entire secret required to perform the cryptographic  
636 operation, then the leakage of information from only one “party” would not enable a  
637 successful attack on the original secret.

638 However, when all these parties reside on a single chip, we must assume that an attacker  
639 can gain *some* (bounded) information about *every* party. In that case, it may happen that the  
640 threshold cryptosystem only complicates a side-channel attack by a small factor, depending  
641 on the number of parties. For example, the  $n$ -out-of- $n$  threshold block cipher by Brickell et  
642 al. [BCF00] uses the  $n$ -fold composition (or cascade) of a block cipher with  $n$  different keys,  
643 which may slow down power analysis attacks only by roughly a factor of  $n$ .

644 Nevertheless, there exist sound countermeasures against side-channel attacks where the  
645 secret variables are split into shares, such that a threshold number of shares can be used to re-  
646 combine the secret, but fewer shares reveal no information at all. We described the theoretical  
647 foundation of these approaches and their resistance against side-channel attacks in Sec. 2.

## 648 4 Models

649 The basic security model for conventional cryptographic algorithms assumes an ideal black  
650 box, in which the cryptographic computations are correct and all internal states, including  
651 keys, are kept secret. Such ideal constructs would not leak any secret information through  
652 side-channels, such as timing and power. In other words, in the ideal black-box the time  
653 and energy used for operations would be independent of secrets, e.g., being instantaneous or  
654 requiring constant time. Under this assumption, one can reduce the problem of evaluating the  
655 algorithm’s security properties to the complexity of the best-known attack against this model.

656 For example, one can define the security strength, which can also be expressed as bit  
657 strength, of different classes of cryptographic algorithms based on the amount of work  
658 needed to perform a brute-force search of the key in a large space related to the key  
659 size. When the algorithms are implemented in real hardware and software, the black-box  
660 assumption can break down in several ways. For example, bugs in the implementation can  
661 lead to side effects that compromise the secret key, as with Heartbleed. Also, the material



662 and electromagnetic characteristics of the platforms on which the algorithms run can cause  
663 side-channel information to leak and allow attackers to recover the secret key.

664 The distinction of ideal versus real implementations can yield useful insights into  
665 the assessment of threshold schemes for cryptographic primitives. What are the security  
666 advantages and disadvantages of performing separate computations on shares of a key,  
667 compared to conventional implementations that use a single secret key? How can threshold  
668 cryptography mitigate the potentially disastrous consequences that a coding error or a  
669 side-channel leak could have on a conventional implementation?

670 This section considers how a range of applicable scenarios may differently affect a range  
671 of tradeoffs between several security properties. These scenarios depend on adversarial goals  
672 and capabilities, and various properties of the system model. It is important to be aware that  
673 security strengthening and weakening may co-exist. The discussion also precludes the next  
674 section, which motivates the need to describe characterizing features of threshold schemes.

#### 675 **4.1 Security considerations**

676 In a first baseline comparison, a real implementation allows vectors of attack not possible  
677 in an ideal black-box. Once these are identified, one asks how to augment conventional  
678 implementations, in the real world, to improve security. Particularly, *how does a threshold*  
679 *approach affect security, compared to a non-threshold approach?* Perhaps security is  
680 improved if an attacker is limited to not compromising more than  $f$ -out-of- $n$  components  
681 within a certain time interval. Also, as explained in Sec. 3.2, a threshold design may  
682 make it inherently more difficult to exploit existing compromises (such as noisy leakage)  
683 in the set of “parties”. While these intuitions are valuable, we want to enable a more  
684 meaningful formulation and/or validation of security assertions about implementations  
685 based on threshold schemes.

686 Two general metrics of interest are *reliability* and *availability* [Rad97]. We can call  
687 them meta-metrics, since we are specially interested in considering them to measure (even  
688 when just qualitatively/comparatively) the upholding of concrete security properties related  
689 to implementations under attack. Reliability — probability of not failing a security goal  
690 — is specially suited for cases of “all-or-nothing” security, where the break of a certain  
691 property represents a catastrophic failure. For example, if a secret decryption key is leaked,  
692 then secrecy is lost with respect to the plaintext associated with public ciphertexts, without  
693 anything being able to revert it. Availability — proportion of time during which a security  
694 goal is satisfied — can be used to measure the actual “availability” of a service or property,  
695 e.g., the proportion of cryptographic output produced as intended. These metrics also depend  
696 on the mission time of an application, so it is relevant to consider, for example, resilience  
697 enhanced by *rejuvenating* compromised components back into a healthy state.

698 **Diverse security properties.** A threshold augmentation may have different effects across  
699 different security properties, e.g., confidentiality vs. availability vs. integrity, possibly  
700 improving one while degrading others. To show the nuances, consider the threshold RSA-  
701 signature scheme described in Sec. 3.1, supported on a 3-out-of-3 secret sharing of the  
702 key. There, each node loses visibility of the original signing key, but retains the ability to  
703 influence the output of a computation dependent on the key. If a compromised node simply  
704 refrains from outputting, then it compromises the availability of the signing operation. If a  
705 corrupted node outputs a syntactically valid but semantically incorrect output share, then it  
706 may or may not compromise integrity, depending on whether or not the mechanism (implicit  
707 in the example) responsible for recombining the output shares is prescribed or not to verify  
708 the correctness of the signature. In summary, for the example scheme considered, there are  
709 different compromise thresholds for different properties:  $f_C = 2$  for confidentiality;  $f_A = 0$   
710 for availability;  $f_I = 0$  or  $f_I = \infty$  (depending on the protocol) for integrity.

711 It is thus conceivable that, under certain types of attack, the threshold scheme may, in  
712 comparison with the conventional scheme, improve the confidentiality of the original key,  
713 while degrading the availability and/or integrity of the intended output. Particularly, this  
714 happens if: compromising the integrity or availability of **one** ( $= 1 + f_A$ ) out of the three  
715 nodes in the threshold version is easier than compromising the availability of a conventional  
716 non-threshold version; (when  $f_I = 0$ ) if compromising the integrity of **one** ( $= 1 + f_I$ ) out  
717 of the three nodes in the threshold version is easier than compromising the integrity of a  
718 conventional non-threshold version; if compromising the confidentiality in the conventional  
719 implementation is easier than compromising the confidentiality of **all**  $n$  ( $= 1 + f_C$ ) nodes in  
720 the threshold version. In some attack/compromise models it may be possible to quantify  
721 the likelihood of  $f + 1$  nodes being compromised, e.g., dependent on an attack intensity  
722 and rejuvenation pattern [BB12]. In particular, one may find that under certain models the  
723 threshold property induces less reliability or availability, e.g., if not properly provisioned  
724 with rejuvenation techniques.

725 Consider the mentioned case with threshold  $f_I = 0$  for integrity. In a context where  
726 integrity is as important as confidentiality, can the above mentioned scheme still be appro-  
727 priate? Yes, since the difficulty of compromising each property may vary with the conceived  
728 type of attack on the implementation. For example: compromising confidentiality may be  
729 possible by *passively* exploiting side-channel leakage from a set of nodes; compromising  
730 integrity may require actively intruding a node to (maliciously) change an internal state (e.g.,  
731 an incorrect share). Particularly, a security property  $P_1$  having a compromise threshold value  
732  $f_1$  lower than the threshold  $f_2$  of another property  $P_2$  does not imply that  $P_1$  is easier to break  
733 than  $P_2$ . Thus, there may be scenarios justifying a threshold scheme with a high threshold  
734 for some properties, even if with a low threshold (including  $f = 0$ ) for others. Properties  
735 with associated threshold 0 may possibly be distinctively protected per node, e.g., based on  
736 standard non-threshold techniques, or be dealt with at a different application layer.



737 **A word of caution: pitfalls of decoupling security properties.** A simplistic decoupling  
738 of security properties may lead to pitfalls. An enumeration of separate security properties  
739 (e.g., privacy of input and correctness of output) may sometimes fail to capture relevant  
740 dependencies or other independent properties. A typical example in cryptography research  
741 is related to commitment schemes, useful for auction applications as follows: first, each  
742 agent independently commits to a chosen bid, in a way that *hides* its value but *binds* the  
743 agent to the value; then all agents reveal their bids in a verifiable way, and the one with  
744 the highest bid wins. An over-simplistic analysis of the application could determine that  
745 the commitment would only need to ensure *hiding* and *binding* properties — respectively  
746 mappable to confidentiality and integrity properties. However, this would fail to capture a  
747 needed property of *non-malleability* [DDN03]: upon seeing a commitment from someone  
748 else, an agent should not be able to produce a new commitment that commits to a value  
749 related to the originally committed value, and which the agent is able to open upon seeing  
750 the opening of the original commitment. There are hiding-and-binding commitments that are  
751 simultaneously malleable [Ped92], which would be ill-suited to the mentioned application.

752 In contrast to the mentioned pitfall, there are formal methods for defining and proving  
753 security. For example, the ideal-real simulation paradigm [Can01] provides an abstraction  
754 that captures the intended application in an ideal world. Starting with such modeling,  
755 one can then deduce diverse properties, such as confidentiality, integrity and availability,  
756 among others (e.g. non-repudiation, or plausible deniability). If some intended property  
757 is not present, then the specified ideal world is not capturing the intended functionality,  
758 and perhaps a different ideal version should be specified. This formal approach may offer  
759 useful properties, such as composability, allowing upper layer protocols to be analyzed by  
760 replacing the threshold protocol by a corresponding ideal functionality.

761 **Specific attacks.** As just conveyed, there is a phase of security assessment that justifies  
762 care about pitfalls of basing the analysis on a limited number of security properties. In that  
763 regard, we assume as baseline that a conventional' implementation already implicitly satis-  
764 fies the security requisites of an intended context. For example, if we discuss a block-cipher  
765 or a signature algorithm, then we assume we are talking of corresponding algorithms already  
766 suitable under some formal model. In other words, the reference conventional system would  
767 be secure if its implementation was not subject to compromise in the real world. It is then  
768 that we position our perspective about threshold schemes in a setting that considers specific  
769 attack vectors in the real world. These attacks, exploiting differences between conventional  
770 implementations and their idealized versions, may sometimes be focused on specific security  
771 properties, e.g., confidentiality of a key. For possible relations between threshold parameters  
772 (e.g.,  $f$  and  $n$ ), other features (see Sec. 5), and the assumed difficulty to perform exploits (e.g.,  
773 per node), we consider how threshold approaches can affect (e.g., improve) security proper-  
774 ties of interest. This may include asking how difficult it is to compromise more than  $f$  parties,  
775 and/or to extract meaningful information from leakage collected from a threshold scheme.  
776 To be clear, this is not incompatible with threshold schemes being themselves provably

Axis	Representative question
passive vs. active	Does the attack affect the specified protocol flow?
static vs. adaptive	To which extent are the attacker's choices based on observations of the protocol execution?
invasive vs. non-invasive	Does an attack require physical access to and/or does it affect the physical structure of a device?
communication interfaces vs. side-channels	Is the attack based on information channels not modeled in the protocol specification?
detectable vs. undetectable	Is the system aware of (e.g., reacts to or logs evidence of) attempted attacks and/or successful intrusions?
threshold-related vs. similar between non-threshold and nodes	Is an attack to the threshold scheme a straightforward generalization (e.g., parallel or sequential attack to nodes) of a possible attack to the conventional implementation?

**Table 1.** Representative attack types

777 secure within formal models of security, e.g., within the ideal/real simulation paradigm. Our  
 778 focus is in asking how and which threshold schemes may improve security in the real world.

## 779 4.2 Types of attack

780 Security goals are considered with respect to an adversary, also known as an “attacker”.  
 781 When evaluating a proposal for threshold scheme implementation, we would like to have  
 782 a sense of the range of adversarial scenarios that it may be able to withstand. As a baseline  
 783 to crosscheck security assertions, we consider several attack types, as enumerated in Table 1.  
 784 This is not intended as a full characterization or rigorous taxonomy, but it helps us recall  
 785 and differentiate relevant cases when considering threshold schemes.

786 **Passive vs. active.** A passive attacker (or a passively corrupted node) does not change  
 787 the flow of the prescribed protocol execution, but may gain knowledge of the internal state  
 788 of some participants, as well as read the content transmitted via communication channels.  
 789 In active attacks, some components may be subject to intrusion and behave arbitrarily  
 790 differently from the protocol specification; in the later case, the attacker may also interfere  
 791 with the communication channels, by altering, dropping and/or reordering messages.

792 **Static vs. adaptive.** In static attacks, the attack pattern, e.g., the choice of which compo-  
 793 nents to try to compromise, does not depend on observations of the protocol execution. In  
 794 adaptive attacks, the attacker can adapt the adversarial actions based on an observation of

795 the protocol flow. For example, a node may be targeted for intrusion upon being elected to  
796 a role of *leader* in a phase of the protocol.

797 **Communication interfaces vs. side-channels.** Some attacks can be perpetrated via regu-  
798 lar communication channels, though possibly using specially crafted messages. For example,  
799 a corrupted client may send an invalid message to a node of a threshold scheme in order  
800 to exploit a buffer-overflow vulnerability. Other attacks can be based on *side-channels*, as  
801 mentioned in Sec. 3.2, taking advantage of an information flow outside the scope of the  
802 explicitly designated communication interface of the system.

803 **Detectable vs. undetectable.** Attacks may be detectable (and detected or undetected) or  
804 undetectable. The latter may happen due to adversaries that are able to bypass possible  
805 attack-detection mechanisms. They may also result from blatant attacks, if the attacked  
806 system is nonetheless unprepared for detection. When a system does not detect being under  
807 attack or having been compromised, it is unable to initiate reactive measures of attack miti-  
808 gation. It may nonetheless have proactive measures in place, triggered at regular intervals of  
809 time, e.g., replacing components that might or might not meanwhile have been compromised.  
810 The prospect of attack detectability may also act as a deterrent against malicious behavior.  
811 From a different angle: a stealth attack may lead to a detectable compromise/intrusion; a  
812 detectable attack may lead to an undetected compromise/intrusion.

813 **Invasive vs. non-invasive.** Another attack characterization relates to the needed proximity  
814 and interaction between the attacker and the physical boundaries of the attacked system.  
815 Non-invasive attacks do not require interaction within the physical boundary of the sys-  
816 tem [ISO12]. Invasive attacks require the attacker to be in presence of (e.g., “touching”)  
817 the physical device or be in its immediate proximity. This includes the case of stripping out  
818 some coating layers of a device, to reach an area of a circuit that can then be directly probed.  
819 This may also include beaming ultra-violet light into particular zones of a circuit (which  
820 requires close proximity), to change an internal state (e.g., a lock bit [AK96]) and thereby  
821 inducing a change of behavior.

822 **Conventional vs. threshold-related.** While threshold schemes may be designed to miti-  
823 gate the effectiveness of some attacks on conventional applications, the actual implementa-  
824 tion of a threshold design may be the cause of new inherent vulnerabilities. For example,  
825 an attack may be able to exploit some vulnerability in the communication network that  
826 intermediates several nodes, where such a network would not even exist in a conventional  
827 implementation. We characterize an attack as threshold-related if the attack vector is in-  
828 herently allowed by the threshold design. Complementary, there are conventional attacks  
829 that can be considered similarly with respect to each component of a threshold scheme. In

830 the latter case, it is still relevant to consider, for example, if an attacker is able to choose  
831 whether to attack the nodes/platform in parallel or sequentially.

832 Tolerance to compromise can be useful even in scenarios of non-intentional adversaries.  
833 For example, some systems may be constrained to satisfy auditability requirements that  
834 warrant taking down components for audit. If a service is supported on a multi-party  
835 threshold scheme with tolerance to compromise, then the audit of components can be done  
836 without affecting the overall availability.

### 837 4.3 System model

838 The goal of this subsection is to convey possible nuances of system models, in order to  
839 encourage a reflection of different consequences they may induce. Several characterizing  
840 features of system model for threshold schemes are further discussed in Sec. 5.

841 **Interactions.** For a security assessment, it is relevant to consider the interaction between  
842 the threshold system and its environment. A threshold system, e.g., a module composed of  
843  $n$  nodes, usually interacts with its clients/operators, through a medium of communication.  
844 The system may also include other interfaces through which a (possibly stealthy) adversary  
845 may obtain information and/or actively interact with components of the system. Thus, attack  
846 vectors are not limited just to actual intrusion/compromise of nodes, but also to adversarial  
847 effects on the environment. For example: corrupted clients may behave maliciously to try  
848 to induce a denial of service for other clients; an adversary controlling part of the network  
849 might be able to induce a state of inconsistency across different nodes, even if no node in  
850 particular can be said to be compromised. We are interested in security properties involving  
851 both the threshold entity and the complementary environment.

852 Besides the  $n$  nodes and users/clients, there may also exist special auxiliary components  
853 with the task of relaying, proxying and/or aggregating messages. Such components, which  
854 we may call *brokers*, can conceivably be outside of the threshold compromise model (i.e.,  
855 not accounted in  $n$ ). Particularly, it may be justifiably assumed that a broker does not fail  
856 within the attack model considered for the other components. For example, a broker may  
857 be a simple stateless web-redirector, independent of the cryptographic computation needed  
858 by the threshold components. Conversely, the  $n$  nodes accounted for the threshold may be  
859 instantiated in a platform more susceptible to certain attacks.

860 A broker can be used to modularize some concerns, e.g., replacing or substantiating  
861 usual assumptions, such as the existence of authenticated channels. Depending on the  
862 communication model, the broker can, for example, broadcast messages from clients to  
863 all components. At the inter-node level, the broker can be a router at the center of a star  
864 configuration, substantiating an inter-node (logical) clique model. The broker can also act as  
865 a mediator between each client and the set of nodes of the threshold scheme, possibly hiding

866 from the client the threshold scheme layer. For example, the broker can produce secret shares  
867 of the client's messages and then only send these shares to the nodes; in the reverse direction,  
868 it can check consistency, and possibly perform error correction, and aggregate replies from  
869 a threshold number of nodes, to then just send a consolidated reply to the client. Depending  
870 on the protocol, the threshold nature can be hidden or not from the client. Even in the broker  
871 case, the threshold nature of the scheme may, as a feature, be intentionally revealed to the  
872 client. For example, the client may receive a multi-signature enabling non-repudiation of  
873 the participation of a number of nodes in the production of a response.

874 The security of a cryptographic service also depends on the communication model. Con-  
875 ceivably, an attacker may be able to eavesdrop, delay, drop, corrupt and/or forge messages  
876 in a number of communication channels. A protocol secure in the case of synchronous,  
877 fail-safe (messages always delivered) and authenticated channels, may become insecure if  
878 the channel conditions change. Thus, the characterization of the communication model is  
879 essential to contextualize security claims about a threshold scheme. Main characterizing  
880 parameters include the existence or lack of synchrony, authentication and encryption. Also,  
881 the presence of certain trusted components (or trusted setups) may significantly affect the  
882 capabilities of the system. For example, the existence of trusted clocks may sometimes  
883 be sufficient to counteract certain difficulties imposed by asynchronous communication  
884 channels. It is specifically pertinent to justify when transport layer security (TLS) should be  
885 or not be required for communication.

886 **Identity trust.** It is easy to leave implicit certain assumptions about the identities of nodes  
887 involved in a threshold scheme, but different settings lead to different results. Who decides  
888 and enforces who the participants (nodes) of a multi-party threshold scheme are? Is the  
889 identity of each party verifiable by other parties? Is the set of parties constant, does it change  
890 in a well-defined manner, or is it arbitrarily open to new membership?

891 In an easy scenario, no new nodes join after the onset of a threshold scheme, and their  
892 identities remain valid throughout their lifetimes. A *dealer* knowing a secret can define  
893 the setup configuration, deploying nodes, establishing their identities and possibly even the  
894 inter-node communication channels. The dealer then distributes shares of the secret and  
895 delegates the threshold execution of some cryptographic primitive.

896 A threshold scheme may also be implemented in a setting where the nodes have identities  
897 tied to public keys within a public-key infrastructure (PKI). The PKI can then support secure  
898 authentication and communication (e.g., with confidentiality and integrity of content and  
899 origin) between any pair of nodes. (This assurance assumes that the attacker may control  
900 the delivery of messages between nodes but cannot prevent nodes from accessing the root  
901 certification authority.) With PKI-based signatures, a threshold scheme can be designed to  
902 enable external users to verify that results were indeed obtained upon a threshold interaction.

903 In a different setting, the initial state of parties might be defined by a joint protocol, e.g.,

904 a distributed key generation [Ped92]. The joint computation may yield to every node a share  
905 of a new secret, possibly along with authentication credentials. This can conceivably be used  
906 by a certification authority to generate a new signing key, without ever having it available  
907 (for leakage) in any localized point. In such case, there is no use for a trusted dealer of  
908 shared secrets, although the nodes may still have been deployed by a centralized authority.

909 Some systems may need or benefit from being dynamic with respect to the number  
910 of participants in a protocol. This may involve allowing different parties to dynamically  
911 enter the protocol, thereby making the threshold parameters  $f$  and  $n$  variable (perhaps while  
912 maintaining a fixed  $f/n$  ratio). What if there is no verifiability criterion for the legitimacy  
913 of a new intended guest participant? In a Sybil attack [Dou02] a single entity can forge  
914 multiple entities perceived as valid, thereby easily breaking any fixed threshold ratio  $f/n$   
915 ( $< 1$ ) of compromisable components. Some mitigation measures may involve enforcing a  
916 cost of participation per party, e.g., performing some cryptographic puzzle [JB99].

917 In more controlled settings, there may be a requirement that new parties be able to  
918 prove belonging to an allowed group. This may be based on a PKI certificate signed by  
919 an authority. Some scenarios can justify having a dynamic number of parties in an actual  
920 threshold scheme for cryptographic primitives. This may happen for example in the case of  
921 an implementation with a system of intrusion detection and proactive and reactive refreshing  
922 of nodes. There may be times when the system must refresh some nodes, and due to a high  
923 rate of reactive refreshing it may temporarily have no additional nodes to join.

924 **Trust between clients and threshold scheme.** We have emphasized the use of threshold  
925 schemes as a way to enhance the protection of secret keys. But when the threshold system is  
926 then used to, say, encrypt or sign messages at the request of a client, is there a concern about  
927 confidentiality of the plaintext? An intention to ensure confidentiality of the plaintext may  
928 dictate restrictions on the type of threshold scheme and system model. If the plaintext is to  
929 remain secret, then the client cannot simply send the plaintext in clear to one or several of  
930 the nodes. Alternatively, it may for example: (i) send it through a trusted proxy that creates  
931 and sends a corresponding plaintext share to each node; or (ii) it may communicate directly  
932 a share to each node; or (iii) it may encrypt shares for each node but send them through  
933 a single primary node. Each example may be supported by a nuanced system model, e.g.,  
934 respectively (i) the existence of a special trusted component; (ii) a communication model  
935 where each client can directly communicate with each node; (iii) a PKI (or shared symmetric  
936 keys) enabling encrypted communication with each node.

937 We can also consider the assurances that a client would like to receive from a threshold  
938 scheme operation. We already referred to the possibility of a client receiving independent  
939 signatures (or multi-signatures) from the nodes. Going further, we can also think of clients  
940 wanting to obtain assurance of correct behavior by the nodes. This can be achieved, for exam-  
941 ple, with the support of publicly verifiable secret sharing (PVSS) schemes [Sta96, Sch99].



942 **Distributed agreement/consensus.** To explain the importance of defining a system model,  
943 we use the distributed agreement/consensus problem — fundamental in the area of distributed  
944 systems — to illustrate how varying models can lead to a wide variability of results. This is  
945 a relevant problem for threshold schemes, namely for certain multi-party implementation  
946 settings. The goal of *consensus* is to ensure that all good parties within a group of  $n$  parties  
947 agree on a value proposed by one of the good parties, even if up to  $f$ -out-of- $n$  parties  
948 are compromised. For example, this may be necessary for letting a multi-party system  
949 decide which cryptographic operations to perform in which order, when the system receives  
950 concurrent requests, possibly maliciously delivered, from multiple users.

951 Results relating  $n$  and  $f$  within this setting include many impossibilities [Lyn89], with  
952 myriad nuances depending on communication and failure models. In one extreme, the  
953 problem is unsolvable deterministically in a completely asynchronous setting [FLP85], even  
954 with (non-transferable) authentication and a single crash-stop process (which can only fail  
955 by crashing). Yet, realistic subtle nuances of the system model circumvent the impossibility.

956 For example, the problem is solvable even with Byzantine faults if the processes have  
957 access to randomness [BO83, Rab83] or synchronous communication [PSL80, LSP82,  
958 DDS87]. In those cases the number of good components must be larger than two-thirds of  
959 the total, i.e.,  $k \geq (2n + 1)/3$ , or equivalently  $n \geq 3f + 1$ . If nodes only fail by crash, then a  
960 non-crashed simple-majority is sufficient, i.e.,  $k \geq f + 1$ , or equivalently  $n \geq 2f + 1$  [Lam06].  
961 In another extreme, consensus is solvable even with a single good party if a suitable trusted  
962 setup can be instantiated to enable transferable message authentication. This is the case  
963 when a PKI setup enables cryptographic signatures [PSL80], or in some other setups (e.g.,  
964 reliable broadcast and secret channels in a precomputation phase [PW92]).

965 The discussion above motivates reflecting also on the property of brittleness [Vas15].  
966 This expresses a degree of susceptibility to a breakdown of the security properties (e.g.,  
967 exfiltration of a key) of a particular algorithm due to errors in the configuration and/or  
968 input parameters. In other words, one is concerned with the fragility of a system with  
969 respect to changes in the system model or expected setup. Even if a system has all desired  
970 properties under a well-defined model, it may be unsuitable for real deployment if it fails  
971 catastrophically under reasonable variations of the environment. One would typically prefer  
972 instead some kind of graceful degradation.

## 973 **5 Characterizing features**

974 We now provide a high-level structured review of characterizing features of threshold  
975 schemes, to facilitate the discussion towards criteria for evaluation of concrete proposals.  
976 We intend to motivate a characterization that helps clarify security tradeoffs when reflecting  
977 on diverse adversarial models. Put differently, we find that the upfront clarification of certain  
978 high-level features is important for discussing the standardization and validation of threshold  
979 cryptographic schemes — see Table 2.

Feature	Representation	Examples
Threshold type	Threshold numbers of bad ( $f$ ) and good ( $k$ ) nodes	$\max f = 0, \dots, n/3 - 1, n/2 - 1, n - 1$ or $\min k = n, \dots, 2f + 1, f + 1, 1$
	Variation with security property and attack vector	$(k_{\text{Secrecy}}, k_{\text{Integrity}}) = (1, n)$ , $((n - 1)/2, (n - 1)/2), \dots, (n, 1)$
	Compromise across nodes	common; independent; sequential
Communication interfaces	Client $\leftrightarrow$ crypto module	broadcast; primary node; secret-sharing
	Inter-node structure	star; clique
	Channel protection	TLS
Target executing platforms	Multiple parties vs. single device	multiple interacting computers; multi-chip in single device; threshold circuit design
	Software vs. hardware	crypto libraries; VMs as components; HSM; crypto accelerators
	Auxiliary components	global clock; proxy; RNG; combiner
Setup and maintenance	Bootstrap support	dealer; SMPC
	Rejuvenation modes	reactive vs. proactive; parallel vs. sequential
	Diversity	offline pre-computation vs. on-the-fly; unbounded vs. limited set

**Table 2.** Characterizing features of threshold schemes

## 980 5.1 Threshold values

981 **A threshold.** From within a total number  $n$  of components, a “threshold” can be expressed  
 982 in two ways: a minimum required number  $k$  of *good* (i.e., non-compromised) components;  
 983 or a maximum allowed number  $f$  of *bad* (i.e., compromised) components. This dual  
 984 characterization is useful and we will use it.

985 The considered type of compromise may vary, but we start by focusing simply on  
 986 threshold numbers. In some cases, a design goal is directly set as the ability to withstand the  
 987 compromise of up to a threshold number  $f$  of components. In other cases, design constraints  
 988 such as cost may directly limit the total number  $n$  of components, which in turn may impose  
 989 a threshold number  $k$  of good components, depending on the protocol and adversarial model.

990 **Relating  $n$  vs.  $f$  and  $k$ .** When analyzing proposals for concrete threshold schemes, we  
 991 intend that the system model be sufficiently characterized to enable determining allowed  
 992 relations between  $n$  vs.  $f$  and  $k$ . We now compare two examples that illustrate how thresholds  
 993 can have an extreme variation across security properties.

994 In Sec. 4.1 we already showed how a signature scheme, based on a simple  $n$ -out-of- $n$



995 secret sharing scheme, can have an optimal threshold for confidentiality ( $f = n - 1$ , i.e.,  
996  $k = 1$ ) and at the same time a pessimal threshold for integrity ( $f = 0$ ).

997 For another example, consider a threshold randomness-generator, intended to output  
998 uniformly random bit-strings, periodically or upon request. In a particular specification,  
999 the output randomness can be defined as the XOR of bit-string contributions from several  
1000 generators of randomness (the components of the threshold scheme). The output is then  
1001 uniformly random if at least one (good) contribution is a uniformly random bit-string that  
1002 is independent of the other contributions. Note that the guarantees for independence are  
1003 important but out of scope for this report. Thus, this scheme has an optimal threshold,  
1004 i.e.,  $(k, f) = (1, n - 1)$ , with respect to guaranteeing the desired uniformly random property  
1005 of a produced output. However, if an output generation requires the participation of all  
1006 components, then the scheme also has the worst threshold for availability, i.e.,  $(k, f) = (n, 0)$ ,  
1007 since a single bad party can boycott the output.

1008 The two examples above differ with respect to which properties are optimal vs. pessimal.  
1009 The integrity threshold was pessimal in the first example and optimal in the second one.  
1010 Alternatively, confidentiality is a relevant property with optimal threshold in the first example,  
1011 whereas it is not even considered in the second example. The threshold symbols  $k$  and  $f$   
1012 could be indexed by the corresponding security property (e.g.,  $f_C$  vs.  $f_I$  vs.  $f_A$ , respectively  
1013 for confidentiality, integrity and availability), but we omit indices when the context is clear.

1014 **Different thresholds for the same scheme.** We gave examples for how the same thresh-  
1015 old scheme may be characterized by different thresholds for different security properties.  
1016 Going further, the thresholds may vary even for a fixed qualitative property (e.g., confiden-  
1017 tiality, or integrity, or availability). Typically, an active/malicious/byzantine adversary induces  
1018 a lower fault-tolerance threshold (i.e., lower tolerance to compromise), when compared to  
1019 a passive and/or crash-only adversary. The same is true for system model assumptions, such  
1020 as asynchrony vs. synchrony of communication channels, and the absence vs. existence of  
1021 a trusted setup such as a public-key infrastructure. The distributed consensus problem in  
1022 Sec. 4.3 shows how a threshold can widely vary depending on the setting.

1023 The determination of relevant threshold values can also depend on the primitives used  
1024 and the application context, e.g., how the actual threshold scheme is used in connection with  
1025 other entities. In some applications, a client can check the validity of signatures obtained  
1026 upon request to a threshold signature module. If a detection of an incorrect signature allows  
1027 a proper reaction, then a threshold signature scheme can be useful even if its integrity  
1028 does not tolerate compromised components (i.e., if  $f = 0$ ). One could then argue that the  
1029 application itself allows a different threshold for integrity. Similar verifiability with respect  
1030 to decryption, or symmetric-key encryption, may be more difficult/costlier, though not  
1031 impossible. In fact, certain threshold schemes can be directly built with a property (often  
1032 called robustness) that prevents integrity violations when up to a threshold number of parties  
1033 misbehave. For example, this can be based on verifiable secret sharing schemes, which

1034 allow verification of correct use of shares. It can also be based on zero-knowledge proofs of  
1035 correct behavior.

1036 In the simplest form, a threshold  $f$  is a number that defines a simple partition of subsets,  
1037 distinguishing the set of subsets with more than  $f$  nodes from the remaining subsets. It is  
1038 worth noticing that the concept can extend to more general partitions [ISN89, HM00].

### 1039 **Representative questions about a proposed scheme.**

- 1040 1. For the desired security properties, what are the threshold values ( $f$  and/or  $k$ ), as a  
1041 function of the total number  $n$  of components?
- 1042 2. What envisioned application contexts justify a high threshold for some properties at  
1043 the cost of a low threshold for other properties (or of other mitigation measures)?
- 1044 3. How do threshold values vary with respect to conceivable variations of the system  
1045 model (e.g., synchrony vs. asynchrony, passive vs. active adversaries)?

## 1046 **5.2 Communication interfaces**

1047 The augmentation from a conventional cryptographic implementation to a threshold scheme  
1048 impacts the communication model. Conceivably, a client can now communicate with more  
1049 than one component (hereafter “node”), and the nodes can communicate between themselves.  
1050 In Sec. 4.3 we already described several nuances of system model, including synchrony vs.  
1051 asynchrony, and the possible existence of a broker. We now briefly describe three nuances  
1052 of communication structures related to clients and nodes.

1053 **Client to/from primary node.** The client may communicate with the threshold scheme  
1054 via a single contact component. When such component is one of the  $n$  nodes of the threshold  
1055 scheme, we can call it a primary node for communication. It relays to all other nodes the  
1056 communication from the client (e.g., a plaintext), and inversely the result (e.g., a signature).  
1057 For example, it aggregates intermediate results produced by other components, to then send  
1058 a single consolidated reply to the client. With a static primary node, the threshold tolerance  
1059  $f \geq 1$  would not include the case of communication failure of the primary.

1060 **From client to all nodes.** If the client is aware of the threshold scheme, it may be able to  
1061 replicate a request across all components. A possible advantage is ensuring that all correct  
1062 components receive the same request. Correspondingly, the client may also receive replies  
1063 from all (or a threshold number of) components and only then decide on a final result. In a  
1064 different implementation model, the client can perform secret-sharing on the input and then  
1065 communicate one share per component. This can be used to support confidentiality of the

1066 input, e.g., a plaintext to encrypt or sign. At the very least, this prevents components from  
1067 applying corruptions dependent on the plaintext value. In the reverse direction, the client  
1068 can reconstruct (possibly with error-correction) an output from a set of replied shares.

1069 **Inter-node communication.** In typical threshold schemes, the components have to di-  
1070 rectly communicate between themselves. (An exception is when the client is the sole  
1071 intermediary between nodes). The inter-node network structure influences the efficiency  
1072 and security of communication. In a star configuration, a primary node intermediates all  
1073 communication. In a clique configuration (i.e., a complete graph), all nodes are able to  
1074 directly contact any other node. For efficiency reasons, a star configuration may be used for  
1075 most communication and a clique configuration be available for secondary communications.  
1076 A dynamic selection of the primary node (also known as leader) may enable overcoming  
1077 cases of it being compromised [[CL02](#)].

#### 1078 **Representative questions about a proposed scheme.**

- 1079 1. Are clients aware of the threshold nature of the implementation?
- 1080 2. How is the initial request from a client propagated through the set of nodes?
- 1081 3. How can the inter-node communication be compromised?
- 1082 4. How does the client obtain a consolidated reply based on a set of partial results  
1083 produced by a set of nodes?
- 1084 5. How is the logical/physical “boundary” (see FIPS 140-2 [[NIS18c](#)]) of the system  
1085 affected by the existing communication channels?

### 1086 **5.3 Target computing platforms**

1087 To some extent, the implementation platform can be abstracted from some functional  
1088 properties of a threshold scheme. Yet, there are distinctive platform-related aspects relevant  
1089 for security assessment and validation. We elaborate here on three main instances: single-  
1090 device vs. multi-party; software vs. hardware; and auxiliary components. These aspects can  
1091 affect other features and are relevant for the development of validation profiles.

1092 **Software vs. hardware.** Cryptography is implemented on a variety of computing plat-  
1093 forms. In the early days of the modern technological revolution in computing and communi-  
1094 cations, cryptographic algorithms were implemented predominantly in hardware. Examples  
1095 of such embodiments are the secure phone lines between federal offices in the 1970s. Hard-  
1096 ware implementations provide a level of isolation of the sensitive cryptographic keys and

1097 their utilization in processing information, along with storage and management of keys and  
1098 other sensitive parameters.

1099 It is natural to think of the physical boundary of a dedicated circuit board, a dedicated  
1100 chip, a smart card, or USB key. Thus, one can relate that physical boundary to the ideal  
1101 black box boundary introduced in Sec. 4 and formulate a set of security assertions. This in  
1102 fact is the foundation for FIPS 140-2 [NIS01], which was initially developed for hardware  
1103 cryptographic implementations. This standard contains specific security requirements on the  
1104 physical boundary of hardware modules, namely [NIS01, Section 4], which are concerned  
1105 with ensuring the attacker cannot probe the circuitry and extract the keys.

1106 As the adoption of cryptography extended into e-commerce over the Internet, software  
1107 implementations of cryptography emerged and over the years became a widely used embod-  
1108 iment for cryptographic primitives. Software cryptographic implementations on a general  
1109 purpose computer (GPC) are just like any other software component that runs within the  
1110 control of an operating system (OS). GPCs are much more porous (see Sec. 1) and tend  
1111 to provide fewer assurances with respect to the isolation of cryptographic keys and other  
1112 security-sensitive parameters from unauthorized access by other applications running on the  
1113 same GPC/OS platform, or remotely through the network interfaces of the platform. Corre-  
1114 spondingly, these software modules are subject only to a subset of the security requirements  
1115 in [NIS01] and are limited to a lower level of security assurances they can claim to deliver.

1116 Given this historical context, the distinction of hardware vs. software in FIPS 140-2  
1117 comes from the difference in isolation that the approaches provide, and is not directly  
1118 related to the manner in which the computation is performed. Note, for example, that a  
1119 *Hardware Security Module* (HSM) might actually contain an embedded microcontroller that  
1120 performs the cryptographic computation in *software*. Also, some hardware platforms such  
1121 as a Field-Programmable Gate Arrays (FPGAs) can be “reprogrammed,” a property that  
1122 was historically reserved for software implementations. For the sake of readability, we will  
1123 assume a more “traditional” separation between hardware and software, focusing primarily  
1124 on the isolation properties, rather than on different types of computing platforms.

1125 The hybrid approach to cryptographic implementations aims to benefit from the flex-  
1126 ibility in software and the isolation and/or acceleration in hardware. Here a portion of  
1127 the implementation is in software executing on a GPC/OS platform and another portion is  
1128 executing on a dedicated HSM attached to the same GPC. Examples of such modules are  
1129 the Trusted Platform Module (TPM) [Mor11], or the cryptographic extensions of standard  
1130 CPU instruction sets such as the SGX instruction on Intel platforms [Int18], the TrustZone  
1131 technology on ARM processors [ARM18]. These modules can also be used as secure  
1132 sub-components within a hybrid fault model. The “secure” components have a more re-  
1133 stricted mode of compromise (e.g., only by crash), thereby enabling better thresholds for  
1134 byzantine fault tolerance of a distributed system composed also of larger and less secure  
1135 components [VCB<sup>+</sup>13, BDK17].

1136 In some cases, a specific cryptographic primitive is implemented partially in software

1137 and partially in hardware. For example, an asymmetric RSA signature algorithm may be  
1138 implemented in such a way that the modulo exponentiation is executed in hardware but  
1139 the required padding of the data is implemented in software. In other cases, an entire suite  
1140 of fully implemented cryptographic primitives is implemented in an HSM and used by a  
1141 software component through application programming interfaces (API).

1142 The hybrid approach offers important security advantages for implementing crypto-  
1143 graphic primitives and key management in isolation, as well as performance improve-  
1144 ments. For example, a hybrid implementation could potentially mitigate cold-boot at-  
1145 tacks [HSH<sup>+</sup>09], which allows keys to be recovered in seconds or even minutes after it has  
1146 been removed from the device. Cold-boot attacks typically assume that the keys are stored  
1147 in the virtual memory of the operating system, and might therefore be moved into DRAM.  
1148 An HSM could mitigate this attack by ensuring that keys never leave the HSM.

1149 Another reason to delegate the execution of cryptographic primitives to dedicated  
1150 hardware is for performance improvement. An example of this is the AES extension on  
1151 Intel [Gue09] and AMD CPUs [AMD12]. HSMs offer similar acceleration benefits.

1152 **Single device vs. multi-party.** When a threshold scheme is developed to enable tolerance  
1153 to the compromise of several components, it is intuitive to think of a set of interacting  
1154 parties (also known as nodes or devices). For example, a *multi-party* threshold setting can be  
1155 composed of  $n$  computers communicating over the Internet, or  $n$  hardware security modules  
1156 (HSMs) connected via a private network, or  $n$  virtual machines (VMs) running within the  
1157 same hardware machine. The connectivity may be dynamic, with the components being  
1158 possibly replaceable for testing, updating and patching. In a multi-party computation, the  
1159 nodes may be separated by a network, possibly asynchronous, inherently outside of the  
1160 control of the threshold scheme. For testing and validation, the tester/validator might not be  
1161 able to simulate a realistic communication medium between multiple parties.

1162 In contrast to the alluded multi-party systems, we also consider “single device” settings.  
1163 Main distinctive aspects include, typically, a somewhat rigid configuration of components  
1164 and a well-defined physical boundary. If the device is a hardware circuit, then in most  
1165 cases the connections between inner wires and gates are fixed throughout the life of the  
1166 device. However, there are technologies that actually allow even those components to be  
1167 adapted across the lifetime of the device, e.g. FPGA. Communication synchrony between  
1168 components is often expected and achieved. Threshold schemes are applicable to the single-  
1169 device setting by means of an inner threshold design. There, the inputs and outputs of a  
1170 threshold circuit become encodings (e.g, sets of secret shares) of the inputs and outputs of  
1171 the conventional (non-threshold) circuit. For confidentiality, the threshold property may be  
1172 that no isolated subset of up to  $f$  wires in the threshold circuit contains information about  
1173 any bit that would be flowing in the original circuit. A main application of this design is  
1174 providing increased resistance against certain side-channel attacks [NRR06].

1175        There is flexibility in distinguishing, and identifying similarities, between multi-party  
1176 and single-device scenarios. For example, we could imagine the physical components within  
1177 a device with a threshold design to be multiple “parties”. Conversely, a single-device may  
1178 indeed not have any redundancy of hardware components, and yet a threshold scheme be  
1179 applied by means of repeated executions of an algorithm. The value of distinguishing the  
1180 platforms is in facilitating a categorization of aspects that may justify different standard-  
1181 ization and/or validation profiles. For example, in a multi-party setting it may be easier to  
1182 isolate, replace and test corruption of a singular component, for the purpose of validating  
1183 properties of an implementation. In some single-device cases, it may be infeasible to achieve  
1184 complete separation of components to test their individual correctness.

1185 **Auxiliary components.** Threshold schemes may require essential components beyond  
1186 those accounted in  $n$ . To use a distinctive term, we call them *auxiliary* components. These  
1187 may include for example a trusted global clock, a proxy, a common random (or pseudo-  
1188 random) bit generator, a combiner of information from components. Having a threshold-  
1189 scheme characterization that acknowledges these components enables a better system model  
1190 for security assessment. For example: a trusted (assumed trustworthy) clock may be what  
1191 enables synchrony in a system model, which in turn can influence the threshold and the  
1192 protocol; the interaction with a trusted random number generator may be necessary to take  
1193 advantage of the threshold design of a circuit based on secret-sharing; we have also already  
1194 given examples of how the auxiliary components may affect the inter-node and the client-  
1195 node communication interfaces. The auxiliary components may have their own compromise  
1196 model, and their testing and validation is also needed when testing and validating a threshold  
1197 system. Yet, it is foreseeable that a great deal of analysis about the auxiliary components  
1198 can be modularized away from threshold-related arguments.

### 1199 **Representative questions**

- 1200        1. If a proposed threshold scheme is devised for a “single-device” setting, what can go  
1201        wrong if its components are instead separated and communicate over the Internet?
- 1202        2. Which parts of the logical boundary of the threshold system do not correspond to a  
1203        physical boundary, as verified by the system developer or deployer?
- 1204        3. Is the system simply developed at the software layer, or are there software components  
1205        tied to particular hardware components?
- 1206        4. Which auxiliary components support the threshold scheme but have a failure model  
1207        different from the one applied to the threshold nodes?



## 1208 5.4 Setup and maintenance

1209 In some settings a threshold scheme can be implemented from scratch as an alternative to  
1210 a construction with a single point of failure. In other cases the starting point is exactly an  
1211 existing single-point-of-failure entity, which is intended to be redesigned as a threshold  
1212 system. To compare the effects from the change, we should consider how the system is  
1213 bootstrapped, including “who” deploys the nodes, and their initial states. Also relevant is the  
1214 setup of the communication network and continued maintenance of the system, including  
1215 during detection and/or recovery of compromised components.

1216 **Dealer vs. dealer-free setup.** In secret sharing, a “dealer” is an entity, possibly outside  
1217 the failure model of the threshold scheme, that knows a secret and “deals” shares of it to the  
1218 nodes of the threshold scheme. In a possible scenario, a key holder in a safe environment  
1219 deals shares of a long-term signature key to nodes that operate in a threshold manner in  
1220 a less-secure environment. The role of a dealer is not necessarily limited to applications  
1221 related to secret keys. As a practical example, a setup phase can also consist of a trusted  
1222 party generating and secret sharing so-called “Beaver-triplets” — triplets of field elements  
1223 (possibly bits) where the third is the product of the first two. The pre-processing of these  
1224 triplets enables a very-efficient execution of certain secure computation protocols [Bea92].

1225 **Rejuvenation of nodes.** It is desirable that compromising  $f$ -out-of- $n$  nodes in a good  
1226 threshold scheme is not easier than compromising 1-out-of-1 in a conventional scheme.  
1227 But is such property inherently guaranteed if  $f > 0$  and if the process of compromising  
1228 each node is independent? Not necessarily, even if the compromise of a node requires an  
1229 independent exploitation effort (e.g., time, computation) per node.

1230 If nodes of a threshold system can only transition from an uncompromised to a com-  
1231 promised state, then the system may be less secure under certain attack vectors. This may  
1232 be due to an increased attack surface, a sufficiently low  $f/n$  ratio and a sufficiently high  
1233 mission time. This is a well-known result in fault tolerance, as may happen in a basic  
1234 triple-modular-redundancy design [KK07]. One may also consider adversarial scenarios  
1235 that induce a probability rate of a node under attack becoming compromised [OY91]. To  
1236 counteract these transitions, it is possible, and in many cases settings essential, to imple-  
1237 ment recovery/replacement/rejuvenation of nodes that can bring nodes back to a “healthy”  
1238 (uncompromised) state. There is a plethora of possible rejuvenation modes, e.g., reactive vs.  
1239 proactive, parallel vs. sequential, instantaneous vs. delayed, stateless vs. stateful, etc.

1240 If a compromise is detected, then the corresponding node should be reactively replaced  
1241 by a healthy version, lest the system eventually converges to all nodes being compromised.  
1242 If the compromises are not detectable but are nonetheless conceivable, then a proactive  
1243 recovery should take place. In the threshold signature scheme from Sec. 3, the resharing of  
1244 the secret key constitutes a parallel rejuvenation of nodes. If there is no persistent intrusion,

1245 and the number of compromises never exceeds the allowed threshold, then the resharing  
1246 brings the whole system back to a pristine state, with all nodes healthy.

1247 The rejuvenation feature brings along a whole new set of considerations, possibly affect-  
1248 ing security in non-trivial ways. If the nodes need to be stateful (i.e., hold state about the  
1249 application), then newly inserted nodes need to be consistently updated, which requires spec-  
1250 ification as a sub-protocol. The rejuvenation of a previously compromised node may need to  
1251 diversify some component, to prevent re-exploitation of the same vulnerability [KF95]. The  
1252 diversification operation may have its own requirements, possibly requiring pre-computation  
1253 vs. being generated on-the-fly by some sampling procedure.

1254 In some protocols a rejuvenation may have to take place in parallel, e.g., such as the  
1255 already discussed example of updating key shares, with all online parties being rejuvenated  
1256 simultaneously. In other cases, rejuvenations may occur sequentially, replacing/recovering  
1257 each node at a time, specially if the process involves a long down time. Many of the  
1258 considerations pertinent to the initial setup of a threshold system are also applicable to the  
1259 rejuvenation context. For example, is there a “dealer” responsible for setting up the full state  
1260 of a rejuvenated node or should the state be updated by the set of online nodes?

1261 If a threshold scheme is based on electing a primary node, what happens when the  
1262 primary node is the one in need of replacement? If a scheme allows reactive and proactive  
1263 rejuvenations, can an attacker take advantage of knowing the schedule/ordering of the  
1264 proactive rejuvenations? What happens if the regular threshold scheme performs correctly  
1265 in an asynchronous environment, but the recovery procedure requires synchrony? Not  
1266 handling asynchrony in recovery procedures may hide subtle problems [SNV07]. If the  
1267 regular threshold scheme requires only a simple honest majority, but the corresponding  
1268 rejuvenation mechanism requires a 2/3 honest majority, then the threshold properties are  
1269 also affected.

1270 **Levels of diversity.** A main motivation for threshold schemes, as an intuitive way to  
1271 improve security by withstanding the compromise of some nodes.<sup>7</sup> Yet, a standalone charac-  
1272 terization of threshold values does not say anything about the difficulty of compromising the  
1273 threshold number  $f$  of nodes. Consider the case of a common vulnerability, i.e., common  
1274 across all nodes (e.g., a bug in a common operating system). Once the vulnerability is  
1275 discovered, an adversary might be able to exploit it with negligible cost to compromise  
1276 all nodes. In this example, this would then be “as easy” as compromising a conventional  
1277 scheme with the same vulnerability.

1278 Consider an example where all nodes are symmetric with respect to the threshold pro-  
1279 tocol, i.e., all implement the same functionality. One can then imagine all nodes being  
1280 implemented in the same manner, say, the same software, possibly containing a common  
1281 vulnerability. Conversely, each node can also be programmed for the same functionality

---

<sup>7</sup>We also bear in mind the possible mapping of threshold properties into side-channel resistance properties.



1282 via different software versions [CA78]. In practice, common vulnerabilities may occur at  
1283 multiple levels where the set of nodes is homogeneous, e.g., operating system, network pro-  
1284 tocol, hardware design, physical location, password. Diversity may be implemented across  
1285 space (i.e., across the components within a threshold protocol) and time (i.e., replacements  
1286 and executions across time). In the multi-party case, rejuvenation can happen by actually  
1287 replacing a physical node by a new one. In certain single-device settings, rejuvenation  
1288 might be limited to refreshing randomness, while the actual hardware structure remains  
1289 fixed. In a software setting, rejuvenation may correspond to replacing a virtual machine, or  
1290 changing some randomness used when compiling a software version. At some levels, there  
1291 may be a small set of variants, e.g., operating systems, whereas others (e.g., passwords) are  
1292 impossible to replace.

1293 The use of diversity is a longstanding practice of traditional fault-tolerance, but its use for  
1294 security is more intricate [LS04]. Implementation-wise, multiple levels of *diversity* (among  
1295 other properties) may be required to substantiate an assumption that compromising more  
1296 nodes is more difficult than compromising fewer nodes. A fundamental difficulty is that the  
1297 level of effort used by an attack vector may be unpredictable until the attack takes place.

#### 1298 **Representative questions.**

- 1299 • Can a threshold scheme be bootstrapped in both dealer and dealer-free manners?
- 1300 • What levels of diversity are envisioned to deter common-mode failures?
- 1301 • What dependency of compromise exists across nodes, for envisioned attack vectors?
- 1302 • Does the sub-protocol for handling rejuvenations interfere with the system availability?

## 1303 **6 Validation of implementations**

### 1304 **6.1 The existing CMVP and FIPS 140-2**

1305 Governments recognize cryptography's important role in protecting sensitive information  
1306 from unauthorized disclosure or modification, and tend to select algorithms with well-  
1307 established theoretical security properties. For example, US and Canadian federal agen-  
1308 cies must use NIST-defined cryptographic algorithm standards to protect sensitive data  
1309 in computer and telecommunications systems [tC96]. They must also use only validated  
1310 cryptographic implementations, typically referred to as modules.

1311 As we have pointed out, the correct and bug-free implementation of a cryptographic  
1312 algorithm and the environment in which it executes are also very important for security. To  
1313 assess security aspects related to real hardware and software implementations, NIST estab-  
1314 lished the Cryptographic Module Validation Program (CMVP) [NIS18c] in 1995 to validate

1315 cryptographic modules against the security requirements in Federal Information Processing  
1316 Standard (FIPS) Publication 140-2 [NIS01]. The CMVP leverages independent third-party  
1317 testing laboratories to test commercial-off-the-shelf cryptographic modules supplied by  
1318 industry vendors.

1319 FIPS 140-2 is a standard defined as a system of conformance security assertions. The  
1320 security assertions in the standard cover a wide range of cryptographic primitives imple-  
1321 mented into various types of physical embodiments called cryptographic modules. The  
1322 security assertions are grouped into sets, one for each security level. FIPS 140-2 defines four  
1323 security levels for cryptographic modules. Depending on the type of technology used for  
1324 a particular module, e.g. software or hardware, the standard defines a subset of applicable  
1325 security assertions that the module must meet for a chosen security level and module-specific  
1326 functional capabilities. In turn, the cryptographic primitives approved by NIST and adopted  
1327 in FIPS 140-2 through Annex A for use in cryptographic modules are also specified as  
1328 sets of conformance security assertions. This allows the CMVP to work with a reasonably  
1329 constrained and well-defined set of security assertions that can be validated.

1330 The Common Criteria [Com17] follows a contrasting approach, where one is allowed  
1331 to define a unique set of security assertions for a target component, often referred to as a  
1332 target of evaluation (TOE). The goal of the Common Criteria certification then is to evaluate  
1333 the correctness of the specific security assertions claimed by the TOE. The evaluation is  
1334 typically much less structured than the validation process in FIPS 140-2, takes longer time  
1335 and requires substantially higher expertise from the evaluators and validators.

## 1336 6.2 Integration of threshold cryptographic schemes

1337 When we consider standardizing threshold cryptographic schemes for approved NIST cryp-  
1338 tographic primitives, we intend to pursue the approach of conformance security assertions,  
1339 similar to the approach taken for the cryptographic primitives and modules.

1340 FIPS 140-2 already has security requirements for secret sharing applied to cryptographic  
1341 keys. Section 4.7.4 of the standard defines security requirements for split-knowledge  
1342 procedures for security levels 3 and 4, stipulating that “*if knowledge of  $n$  key components*  
1343 *is required to reconstruct the original key, then knowledge of  $n - 1$  components provides no*  
1344 *information about the original key, other than the length.*” This can for example be satisfied  
1345 by implementations of the Shamir and Blakley secret sharing schemes mentioned in Sec. 2.1.

1346 As technology progresses and cryptography becomes ubiquitous in the federal infor-  
1347 mation infrastructure, the number and complexity of modules to be validated increases.  
1348 This makes it increasingly difficult to detect at validation stage all possible defects that  
1349 might compromise security. This is one more reason to consider the potential of threshold  
1350 cryptography in avoiding single points of failure in real implementations. However, similarly  
1351 to conventional cryptography, the security of the threshold cryptographic implementation

1352 may also be impacted by defects introduced as a result human errors or unsafe optimiza-  
1353 tion by the tools used to compile or synthesize the implementation. Thus, it is important  
1354 to ensure that the algorithms supporting threshold cryptography are theoretically secure,  
1355 and to verify that they've been implemented correctly. The definition of guidelines would  
1356 help develop a structured process of formulating and validating security assertions about  
1357 threshold cryptographic implementations.

1358 One additional challenge is to enable ways to validate those assertions in an automated  
1359 fashion. NIST is working with the industry to rebuild its cryptographic validation programs  
1360 and improve the efficiency and effectiveness of cryptographic module testing in order to  
1361 reduce the time and cost required for testing while providing a high level of assurance for  
1362 Federal government consumers. As the NIST cryptographic validation programs evolve,  
1363 the adoption of new cryptographic technology into them should target the future structure  
1364 and mechanisms for testing and reporting results [NIS18b]. The current project includes an  
1365 Industry/NIST collaboration website for automated validation of cryptographic algorithms  
1366 (ACVP) and cryptographic modules [NIS18a, NIS18b].

## 1367 7 Criteria for standardization

1368 Active research over the last few decades has resulted in a substantial body of literature on  
1369 threshold cryptographic schemes. Usually there are tradeoffs of threshold values for different  
1370 security properties, potentially depending on the application context and system model.  
1371 With appropriate caution, threshold cryptography offers a great potential for strengthening  
1372 the security of cryptographic implementations. But what criteria should one use to ask for  
1373 and select from a potential pool of candidate threshold cryptographic schemes?

1374 **Some representative questions.** We intend this document to promote the development  
1375 of criteria for evaluation of proposals of threshold cryptographic schemes. Here we list  
1376 representative questions likely to induce a discussion about this:

- 1377 1. Are the *characterizing features* of the threshold scheme fully described?
- 1378 2. On what *executing platforms* can the scheme be implemented?
- 1379 3. What are the operational costs and properties of *setup and maintenance*?
- 1380 4. What are the *node-rejuvenation* mechanisms (e.g., resharing or node replacement)?
- 1381 5. How *efficient/performant* are the operations as a function of threshold parameters?
- 1382 6. Is the scheme applicable to *NIST-approved* cryptographic primitives?
- 1383 7. Do *base primitives* (e.g., oblivious transfer) require independent standardization?
- 1384 8. Is the *system model* applicable to known and relevant application contexts?
- 1385 9. How is *diversity* of nodes related to known attack vectors?
- 1386 10. Is the implementation complexity likely to lead to *new bugs or misconfiguration*?
- 1387 11. What *trusted setup and assumptions* are required (e.g., dealer, special components)?

- 1388 12. What threshold properties relate to resistance against *side-channel attacks* and how?  
1389 13. Are there identified *security tradeoffs* across attack types and configurations?  
1390 14. Is the security assessment supported by a *security proof*?  
1391 15. How does the *reliability* compare against that of a conventional implementation?  
1392 16. How *brittle* is the scheme (likely to break under small variations in the environment)?  
1393 17. What features of *graceful degradation* exist against conceivable failures?  
1394 18. Do the *security assertions* match / fit into the FIPS 140-2 framework?  
1395 19. How *testable* is the scheme (can security assertions be tested and validated)?  
1396 20. Is there a proposed *automated validation* mechanism?  
1397 21. What are the *intellectual property* implications and the *licensing* conditions?

1398 We need to develop an objective criteria set to support a call for and a selection of  
1399 schemes for standardization. An actual criteria guideline would elaborate further on each  
1400 of the above questions, or variations thereof, and possibly others. The development of such  
1401 criteria would benefit from collaborative public feedback from the cryptography research  
1402 community, as well as from stakeholders in the government and industry.

1403 In addition, there may exist pertinent standardization meta-questions. What flexibility of  
1404 parametrization should a standard allow? Should there be distinct standardization profiles  
1405 to separately focus on distinct attribute instantiations, e.g., single-device vs. multi-party  
1406 platform, side-channel attack vs. intrusion per node? Next, we elaborate a bit further on two  
1407 additional aspects.

1408 **Standardization at what granularity level?** Current industry guidelines for best prac-  
1409 tices in cybersecurity [Ver18] recommend active patching of vulnerable components. If in  
1410 a validated multi-party threshold scheme a node is found to have a serious vulnerability,  
1411 the node may need to be patched. This would not be a problem if the scheme tolerates  
1412 the full compromise of at least one node, and/or if it can replace it with another type of  
1413 (validated) component. In that case, the overall system continues to operate smoothly during  
1414 the patching and revalidation of the vulnerable component. Thus, when considering the  
1415 standardization of a particular threshold scheme, there may be value in validating imple-  
1416 mentations with diverse platforms/implementations for the individual nodes. This example  
1417 suggests a question about the standardization criteria: what levels of granularity/modularity  
1418 should be considered for standardization?

1419 **Standardization opportunities.** Threshold schemes have wide applicability, in the sense  
1420 that there are general techniques to convert a conventional implementation into a threshold  
1421 version thereof. One can thus ask: for which conventional cryptographic schemes should  
1422 one consider standardization of a threshold version? On one hand, there is a clear interest in  
1423 enabling threshold versions of NIST-approved cryptographic primitives. On the other hand,  
1424 the consideration of standardization of threshold schemes is in itself an opportunity to review

1425 suitability for new standards. In this line, we also wonder how the standardization of thresh-  
1426 old schemes might also benefit other ongoing NIST efforts of standardization. For example,  
1427 could elements from the lightweight [MBTM17] and post-quantum cryptography [NIS17]  
1428 projects at NIST be useful for threshold cryptography? Could the schemes considered by  
1429 those projects also be looked at in the perspective of possible threshold variants? We do not  
1430 intend here to show any preference about concrete cases, but simply to raise the point for  
1431 consideration. We believe that a better clarification may arise from a constructive interaction  
1432 with the research community and other stakeholders.

## 1433 **8 Conclusions**

1434 Conventional cryptographic implementations have a single point of failure when the secret  
1435 key is stored in one location, or when a localized fault breaks integrity of the output or avail-  
1436 ability of a cryptographic operation. Threshold techniques can mitigate these failure modes.

1437 For example, secret-sharing schemes can be used to split a secret key while its use is  
1438 not required, ensuring that a threshold number  $f + 1$  of shares is needed to reconstruct the  
1439 key. However, this by itself does not enable cryptographic primitives to use the key shares  
1440 instead of the recombined key. In threshold cryptography the computation is performed on  
1441 shares of the key, without the need to recombine the original key. A simple secret sharing  
1442 might also not enable threshold properties for other goals, such as preventing a corruption  
1443 of the intended output. Threshold cryptography may enable operation modes with threshold  
1444 security for other properties, including integrity.

1445 Generally speaking, we use “threshold cryptography” in the broad sense of encom-  
1446 passing threshold schemes for a secure implementation of cryptographic primitives. This  
1447 includes schemes related to secure multi-party computation and intrusion-tolerant distributed  
1448 systems. Usually, a threshold property is expressed as an  $f$ -out-of- $n$  tolerance to compro-  
1449 mise, where the compromise of up to  $f$  nodes does not break some security property. For  
1450 example, when up to  $f$  parties possess no information about a secret, security against a wide  
1451 range of side-channel attacks can be achieved under some reasonable assumptions about  
1452 the distributions of side-channel information. Furthermore, a threshold scheme may even  
1453 provide resistance against side-channel attacks that collect information correlated across  
1454 all nodes (beyond the threshold). This is because, in some models, a threshold design may  
1455 complicate the exploitation of noisy side-channel information.

1456 Threshold schemes can be efficient. For example, we described how a simple threshold  
1457 RSA signature scheme based on a  $n$ -out- $n$  secret-sharing has a complexity that increases  
1458 only linearly with the number of shares, and whose computation is parallelized across  
1459 several nodes. In such case, the simplicity of the method is based on a mathematical  
1460 structure (a homomorphism) present in the underlying structure of the original scheme.  
1461 In contrast, schemes for other cryptographic primitives, such as some block ciphers, may  
1462 require significant computational overhead compared to their conventional counterparts.

1463 Still, even in those cases the threshold schemes may be practical and justifiable, depending  
1464 on the intended security assurances and the application context.

1465 The discussion in the preceding sections highlighted nuanced security assertions that can  
1466 be obtained about threshold cryptographic schemes. The security of such schemes has to be  
1467 seen through the prism of a security model and possibly considering several system models  
1468 of implementation. For example, there may be differences between active or passive attacks.  
1469 To help navigate the landscape of possible schemes, this report enumerated characterizing  
1470 features and their possible effects. For example: there are potential benefits of rerandomizing  
1471 the shares of the secret key; properties can be different between multi-device vs. single-  
1472 device platforms; some security properties are different depending on the communication  
1473 platform with the environment and between components.

1474 An understanding of a variety of instantiations of characterizing features is necessary  
1475 for the development of objective criteria for selecting candidates for standardization. For the  
1476 purpose of standardization and validation, the combination of characterizing features and  
1477 attack models should be translated into security assertions. The way these can fit into FIPS  
1478 140-2 and/or require complementary standardization is a matter for discussion.

1479 We have looked at numerous factors that influence the type of security assessment  
1480 that can be made about threshold cryptographic schemes. Clearly, threshold cryptography  
1481 has potential to improve the security of the implementation of cryptographic primitives,  
1482 provided it is carefully used. There is a clear interest in enabling threshold schemes for  
1483 already standardized cryptographic primitives. The standardization effort may also constitute  
1484 an opportunity to consider the case for standardizing new primitives. There are long-standing  
1485 research results, and the research community is still active in the area.

1486 We intend this report to initiate a discussion on the standardization of threshold cryp-  
1487 tographic schemes. We can envision some of the challenges ahead. The most immediate  
1488 seems to be the development of criteria for and selection of proposals. This document did  
1489 not put forth such criteria, but motivated the need for one and developed some basis for it.

1490 Once criteria are in place, the selection and standardization of concrete schemes should  
1491 include an integration with validation methodologies. How then to express security asser-  
1492 tions that may fit within FIPS 140-2 or fit well as a complement thereof? What security  
1493 and implementation profiles should be devised? When tackling these challenges, positive  
1494 synergies may result from engaging with and incorporating feedback from the research  
1495 community and other stakeholders.



1496 **References**

- 1497 [ABF<sup>+</sup>03] C. Aumüller, P. Bier, W. Fischer, P. Hofreiter, and J.-P. Seifert. *Fault Attacks*  
1498 *on RSA with CRT: Concrete Results and Practical Countermeasures*. In  
1499 B. S. Kaliski, Ç. K. Koç, and C. Paar (eds.), *Cryptographic Hardware and*  
1500 *Embedded Systems — CHES 2002*, vol. 2523 of LNCS, pages 260–275.  
1501 Springer Berlin Heidelberg, 2003. DOI:[10.1007/3-540-36400-5\\_20](https://doi.org/10.1007/3-540-36400-5_20).
- 1502 [AK96] R. Anderson and M. Kuhn. *Tamper Resistance: A Cautionary Note*. Proc.  
1503 2nd USENIX Workshop on Electronic Commerce (WOEC'96), 2:1–11, 1996.
- 1504 [AMD12] AMD Corporation. *AMD has you covered*. [https://www.amd.com/Documents/](https://www.amd.com/Documents/Security_021.pdf)  
1505 [Security\\_021.pdf](https://www.amd.com/Documents/Security_021.pdf), 2012.
- 1506 [AMGC85] B. Awerbuch, S. Micali, S. Goldwasser, and B. Chor. *Verifiable secret sharing*  
1507 *and achieving simultaneity in the presence of faults*. In Proc. 26th Annual  
1508 Symposium on Foundations of Computer Science, SFCS '85, pages 383–395.  
1509 IEEE Computer Society, 1985. DOI:[10.1109/SFCS.1985.64](https://doi.org/10.1109/SFCS.1985.64).
- 1510 [AMN01] M. Abdalla, S. Miner, and C. Namprempre. *Forward-Secure Threshold*  
1511 *Signature Schemes*. In D. Naccache (ed.), *Topics in Cryptology — CT-RSA*  
1512 *2001*, pages 441–456. Springer Berlin Heidelberg, 2001. DOI:[10.1007/3-540-](https://doi.org/10.1007/3-540-45353-9_32)  
1513 [45353-9\\_32](https://doi.org/10.1007/3-540-45353-9_32).
- 1514 [And02] R. Anderson. *Two remarks on public key cryptology*. Technical report,  
1515 University of Cambridge, Computer Laboratory, 2002.
- 1516 [ARM18] ARM Corporation. *TrustZone*. [https://www.arm.com/products/](https://www.arm.com/products/security-on-arm/trustzone)  
1517 [security-on-arm/trustzone](https://www.arm.com/products/security-on-arm/trustzone), 2018.
- 1518 [BB03] D. Brumley and D. Boneh. *Remote Timing Attacks Are Practical*. In Proc.  
1519 12th Conference on USENIX Security Symposium — SSYM'03, pages 1–13.  
1520 USENIX Association, 2003.
- 1521 [BB12] L. T. A. N. Brandão and A. N. Bessani. *On the reliability and avail-*  
1522 *ability of replicated and rejuvenating systems under stealth attacks and*  
1523 *intrusions*. *Journal of the Brazilian Computer Society*, 18(1):61–80, 2012.  
1524 DOI:[10.1007/s13173-012-0062-x](https://doi.org/10.1007/s13173-012-0062-x).
- 1525 [BCF00] E. F. Brickell, G. D. Crescenzo, and Y. Frankel. *Sharing Block Ciphers*. In  
1526 *Information Security and Privacy – ACISP 2000*, vol. 1841 of LNCS, pages  
1527 457–470. Springer Berlin Heidelberg, 2000. DOI:[10.1007/10718964\\_37](https://doi.org/10.1007/10718964_37).
- 1528 [BDK17] J. Behl, T. Distler, and R. Kapitza. *Hybrids on Steroids: SGX-Based*  
1529 *High Performance BFT*. In Proc. 12th European Conference on Computer  
1530 Systems, EuroSys '17, pages 222–237, New York, NY, USA, 2017. ACM.  
1531 DOI:[10.1145/3064176.3064213](https://doi.org/10.1145/3064176.3064213).

- 1532 [BDL97] D. Boneh, R. A. DeMillo, and R. J. Lipton. *On the Importance of Checking*  
1533 *Cryptographic Protocols for Faults*. In W. Fumy (ed.), *Advances in*  
1534 *Cryptology — EUROCRYPT '97*, vol. 1233 of LNCS, pages 37–51. Springer  
1535 Berlin Heidelberg, 1997. DOI:[10.1007/3-540-69053-0\\_4](https://doi.org/10.1007/3-540-69053-0_4).
- 1536 [Bea92] D. Beaver. *Efficient Multiparty Protocols Using Circuit Randomiza-*  
1537 *tion*. In J. Feigenbaum (ed.), *Advances in Cryptology — CRYPTO '91*,  
1538 vol. 576 of LNCS, pages 420–432. Springer Berlin Heidelberg, 1992.  
1539 DOI:[10.1007/3-540-46766-1\\_34](https://doi.org/10.1007/3-540-46766-1_34).
- 1540 [Ber05] D. J. Bernstein. *Cache-timing attacks on AES*. [https://cr.yp.to/antiforgery/](https://cr.yp.to/antiforgery/cachetiming-20050414.pdf)  
1541 [cachetiming-20050414.pdf](https://cr.yp.to/antiforgery/cachetiming-20050414.pdf), 2005.
- 1542 [Ber06] D. J. Bernstein. *Curve25519: New Diffie-Hellman Speed Records*. In M. Yung,  
1543 Y. Dodis, A. Kiayias, and T. Malkin (eds.), *Public Key Cryptography — PKC*  
1544 *2006*, vol. 3958 of LNCS, pages 207–228. Springer Berlin Heidelberg, 2006.  
1545 DOI:[10.1007/11745853\\_14](https://doi.org/10.1007/11745853_14).
- 1546 [BF97] D. Boneh and M. Franklin. *Efficient generation of shared RSA keys*. In B. S.  
1547 Kaliski (ed.), *Advances in Cryptology — CRYPTO '97*, vol. 1294 of LNCS,  
1548 pages 425–439. Springer Berlin Heidelberg, 1997. DOI:[10.1007/BFb0052253](https://doi.org/10.1007/BFb0052253).
- 1549 [BFM88] M. Blum, P. Feldman, and S. Micali. *Non-interactive Zero-knowledge and Its*  
1550 *Applications*. In Proc. 20th Annual ACM Symposium on Theory of Computing,  
1551 STOC '88, pages 103–112. ACM, 1988. DOI:[10.1145/62212.62222](https://doi.org/10.1145/62212.62222).
- 1552 [BGG<sup>+</sup>14] J. Balasch, B. Gierlichs, V. Grosso, O. Reparaz, and F. Standaert. *On the Cost*  
1553 *of Lazy Engineering for Masked Software Implementations*. In *Smart Card*  
1554 *Research and Advanced Applications — CARDIS*, vol. 8968 of LNCS, pages  
1555 64–81. Springer Berlin Heidelberg, 2014. DOI:[10.1007/978-3-319-16763-3\\_5](https://doi.org/10.1007/978-3-319-16763-3_5).
- 1556 [Bla79] G. R. Blakley. *Safeguarding cryptographic keys*. In Proc. International  
1557 Workshop on Managing Requirements Knowledge, vol. 48 of AFIPS 1979,  
1558 pages 313–317, 1979. DOI:[10.1109/AFIPS.1979.98](https://doi.org/10.1109/AFIPS.1979.98).
- 1559 [BM99] M. Bellare and S. K. Miner. *A Forward-Secure Digital Signature*  
1560 *Scheme*. In M. Wiener (ed.), *Advances in Cryptology — CRYPTO' 99*,  
1561 vol. 1666 of LNCS, pages 431–448. Springer Berlin Heidelberg, 1999.  
1562 DOI:[10.1007/3-540-48405-1\\_28](https://doi.org/10.1007/3-540-48405-1_28).
- 1563 [BN06] M. Bellare and G. Neven. *Multi-signatures in the Plain public-Key Model*  
1564 *and a General Forking Lemma*. In Proc. 13th ACM Conference on Computer  
1565 and Communications Security, CCS '06, pages 390–399. ACM, 2006.  
1566 DOI:[10.1145/1180405.1180453](https://doi.org/10.1145/1180405.1180453).
- 1567 [BO83] M. Ben-Or. *Another Advantage of Free Choice (Extended Abstract):*  
1568 *Completely Asynchronous Agreement Protocols*. In Proc. 2nd Annual ACM  
1569 Symposium on Principles of Distributed Computing, PODC '83, pages 27–30.



- 1570 ACM, 1983. DOI:[10.1145/800221.806707](https://doi.org/10.1145/800221.806707).
- 1571 [Bre12] E. Brewer. *CAP Twelve Years Later: How the "Rules" Have Changed*.  
1572 Computer, 45:23–29, 01 2012. DOI:[10.1109/MC.2012.37](https://doi.org/10.1109/MC.2012.37).
- 1573 [CA78] L. Chen and A. Avizienis. *N-Version Programming: A Fault-Tolerance*  
1574 *Approach to Reliability of Software Operation*. Digest FTCS-8: 8th Annual  
1575 International Conference on Fault Tolerant Computing, pages 3–9, June 1978.
- 1576 [Can01] R. Canetti. *Universally Composable Security: A New Paradigm for*  
1577 *Cryptographic Protocols*. In Proc. 42nd IEEE Symposium on Foundations  
1578 of Computer Science, FOCS '01, pages 136–145. IEEE Computer Society,  
1579 2001. DOI:[10.1109/SFCS.2001.959888](https://doi.org/10.1109/SFCS.2001.959888).
- 1580 [CJRR99] S. Chari, C. S. Jutla, J. R. Rao, and P. Rohatgi. *Towards Sound Approaches*  
1581 *to Counteract Power-Analysis Attacks*. In M. Wiener (ed.), *Advances in*  
1582 *Cryptology — CRYPTO' 99*, vol. 1666 of LNCS, pages 398–412. Springer  
1583 Berlin Heidelberg, 1999. DOI:[10.1007/3-540-48405-1\\_26](https://doi.org/10.1007/3-540-48405-1_26).
- 1584 [CL02] M. Castro and B. Liskov. *Practical Byzantine Fault Tolerance and Proactive*  
1585 *Recovery*. ACM Trans. Comput. Syst., 20(4):398–461, November 2002.  
1586 DOI:[10.1145/571637.571640](https://doi.org/10.1145/571637.571640).
- 1587 [Com17] Common Criteria. *Common Criteria for Information Technology Security*  
1588 *Evaluation*, April 2017. <https://www.commoncriteriaportal.org/>.
- 1589 [CvH91] D. Chaum and E. van Heyst. *Group Signatures*. In D. W. Davies (ed.),  
1590 *Advances in Cryptology — EUROCRYPT '91*, vol. 547 of LNCS, pages  
1591 257–265. Springer Berlin Heidelberg, 1991. DOI:[10.1007/3-540-46416-6\\_22](https://doi.org/10.1007/3-540-46416-6_22).
- 1592 [DDF14] A. Duc, S. Dziembowski, and S. Faust. *Unifying Leakage Models: From Prob-*  
1593 *ing Attacks to Noisy Leakage*. In *Advances in Cryptology — EUROCRYPT*  
1594 *2014*, vol. 8441 of LNCS, pages 423–440. Springer Berlin Heidelberg, 2014.  
1595 DOI:[10.1007/978-3-642-55220-5\\_24](https://doi.org/10.1007/978-3-642-55220-5_24).
- 1596 [DDN03] D. Dolev, C. Dwork, and M. Naor. *Nonmalleable Cryptography*. SIAM  
1597 Review, 45(4):727–784, 2003. DOI:[10.1137/S0036144503429856](https://doi.org/10.1137/S0036144503429856).
- 1598 [DDS87] D. Dolev, C. Dwork, and L. Stockmeyer. *On the Minimal Synchronism*  
1599 *Needed for Distributed Consensus*. J. ACM, 34(1):77–97, January 1987.  
1600 DOI:[10.1145/7531.7533](https://doi.org/10.1145/7531.7533).
- 1601 [DF90] Y. Desmedt and Y. Frankel. *Threshold cryptosystems*. In G. Brassard (ed.),  
1602 *Advances in Cryptology — CRYPTO' 89 Proceedings*, vol. 435 of LNCS,  
1603 pages 307–315. Springer New York, 1990. DOI:[10.1007/0-387-34805-0\\_28](https://doi.org/10.1007/0-387-34805-0_28).
- 1604 [DLK<sup>+</sup>14] Z. Durumeric, F. Li, J. Kasten, J. Amann, J. Beekman, M. Payer, N. Weaver,  
1605 D. Adrian, V. Paxson, M. Bailey, and J. A. Halderman. *The Matter of*  
1606 *Heartbleed*. In Proc. 2014 Conference on Internet Measurement Con-

- 1607                   ference, IMC '14, pages 475–488, New York, NY, USA, 2014. ACM.  
1608                   DOI:[10.1145/2663716.2663755](https://doi.org/10.1145/2663716.2663755).
- 1609        [Dou02] J. R. Douceur. *The Sybil Attack*. In P. Druschel, F. Kaashoek, and A. Rowstron  
1610                   (eds.), *Peer-to-Peer Systems*, pages 251–260. Springer Berlin Heidelberg,  
1611                   2002. DOI:[10.1007/3-540-45748-8\\_24](https://doi.org/10.1007/3-540-45748-8_24).
- 1612    [DSDFY94] A. De Santis, Y. Desmedt, Y. Frankel, and M. Yung. *How to Share a Function*  
1613                   *Securely*. In Proc. 26th Annual ACM Symposium on Theory of Computing,  
1614                   STOC '94, pages 522–533. ACM, 1994. DOI:[10.1145/195058.195405](https://doi.org/10.1145/195058.195405).
- 1615        [Fel87] P. Feldman. *A Practical Scheme for Non-interactive Verifiable Secret*  
1616                   *Sharing*. In Proc. 28th Annual Symposium on Foundations of Computer  
1617                   Science, SFCS '87, pages 427–438. IEEE Computer Society, 1987.  
1618                   DOI:[10.1109/SFCS.1987.4](https://doi.org/10.1109/SFCS.1987.4).
- 1619        [FLP85] M. J. Fischer, N. A. Lynch, and M. S. Paterson. *Impossibility of Distributed*  
1620                   *Consensus with One Faulty Process*. J. ACM, 32(2):374–382, April 1985.  
1621                   DOI:[10.1145/3149.214121](https://doi.org/10.1145/3149.214121).
- 1622        [FNP04] M. J. Freedman, K. Nissim, and B. Pinkas. *Efficient private matching and*  
1623                   *set intersection*. In International conference on the theory and applications  
1624                   of cryptographic techniques, pages 1–19. Springer, 2004.
- 1625        [GMR85] S. Goldwasser, S. Micali, and C. Rackoff. *The Knowledge Complexity*  
1626                   *of Interactive Proof-systems*. In Proc. 17th Annual ACM Symposium  
1627                   on Theory of Computing, STOC '85, pages 291–304. ACM, 1985.  
1628                   DOI:[10.1145/22145.22178](https://doi.org/10.1145/22145.22178).
- 1629        [GMW87] O. Goldreich, S. Micali, and A. Wigderson. *How to Play ANY Mental Game*.  
1630                   In Proc. 19th Annual ACM Symposium on Theory of Computing, STOC '87,  
1631                   pages 218–229. ACM, 1987. DOI:[10.1145/28395.28420](https://doi.org/10.1145/28395.28420).
- 1632        [GRJK00] R. Gennaro, T. Rabin, S. Jarecki, and H. Krawczyk. *Robust and Efficient*  
1633                   *Sharing of RSA Functions*. Journal of Cryptology, 13(2):273–300, Mar 2000.  
1634                   DOI:[10.1007/s001459910011](https://doi.org/10.1007/s001459910011).
- 1635        [Gue09] S. Gueron. *Intel's New AES Instructions for Enhanced Performance and*  
1636                   *Security*. In O. Dunkelman (ed.), *Fast Software Encryption*, 16th International  
1637                   Workshop, FSE 2009, vol. 5665 of LNCS, pages 51–66. Springer, 2009.  
1638                   DOI:[10.1007/978-3-642-03317-9\\_4](https://doi.org/10.1007/978-3-642-03317-9_4).
- 1639        [HJKY95] A. Herzberg, S. Jarecki, H. Krawczyk, and M. Yung. *Proactive Secret Sharing*  
1640                   *Or: How to Cope With Perpetual Leakage*. In D. Coppersmith (ed.), *Advances*  
1641                   *in Cryptology — CRYPTO' 95*, pages 339–352. Springer Berlin Heidelberg,  
1642                   1995. DOI:[10.1007/3-540-44750-4\\_27](https://doi.org/10.1007/3-540-44750-4_27).
- 1643        [HM00] M. Hirt and U. Maurer. *Player Simulation and General Adversary Structures*

- 1644            *in Perfect Multiparty Computation*. Journal of Cryptology, 13(1):31–60, Jan  
1645            2000. DOI:[10.1007/s001459910003](https://doi.org/10.1007/s001459910003).
- 1646    [HSH<sup>+</sup>09] J. A. Halderman, S. D. Schoen, N. Heninger, W. Clarkson, W. Paul, J. A.  
1647            Calandrino, A. J. Feldman, J. Appelbaum, and E. W. Felten. *Lest We*  
1648            *Remember: Cold-boot Attacks on Encryption Keys*. Commun. ACM,  
1649            52(5):91–98, May 2009. DOI:[10.1145/1506409.1506429](https://doi.org/10.1145/1506409.1506429).
- 1650            [IN83] K. Itakura and K. Nakamura. *A public-key cryptosystem suitable for digital*  
1651            *multisignatures*. In NEC J. Res. Dev. 71, pages 1–8, Oct. 1983.
- 1652            [Int18] Intel Corporation. *Software Guard Extention (SGX)*. [https:](https://software.intel.com/en-us/sgx)  
1653            [//software.intel.com/en-us/sgx](https://software.intel.com/en-us/sgx), 2018.
- 1654            [ISN89] M. Ito, A. Saito, and T. Nishizeki. *Secret sharing scheme realizing general ac-*  
1655            *cess structure*. Electronics and Communications in Japan (Part III: Fundamen-  
1656            tal Electronic Science), 72(9):56–64, 1989. DOI:[10.1002/ecjc.4430720906](https://doi.org/10.1002/ecjc.4430720906).
- 1657            [ISO12] ISO. *ISO/IEC 19790:2012, Information technology – Security*  
1658            *techniques – Security requirements for cryptographic modules*.  
1659            <https://www.iso.org/standard/52906.html>, August 2012.
- 1660            [ISO16] ISO. *ISO/IEC 19592-1:2016, Information technology – Security techniques*  
1661            *– Secret sharing – Part 1: General*. <https://www.iso.org/standard/65422.html>,  
1662            2016.
- 1663            [ISO17] ISO. *ISO/IEC 19592-2:2017, Information technology – Security*  
1664            *techniques – Secret sharing – Part 2: Fundamental mechanisms*.  
1665            <https://www.iso.org/standard/65425.html>, 2017.
- 1666            [ISW03] Y. Ishai, A. Sahai, and D. A. Wagner. *Private Circuits: Securing Hardware*  
1667            *against Probing Attacks*. In Advances in Cryptology — CRYPTO 2003,  
1668            vol. 2729 of LNCS, pages 463–481. Springer Berlin Heidelberg, 2003.  
1669            DOI:[10.1007/978-3-540-45146-4\\_27](https://doi.org/10.1007/978-3-540-45146-4_27).
- 1670            [JB99] A. Juels and J. Brainard. *Client puzzles: A Cryptographic countermeasure*  
1671            *against connection depletion attacks*. In Network and distributed system secu-  
1672            rity symposium — NDSS’99, vol. 99, pages 151–168. Internet Society, 1999.
- 1673            [KF95] N. Kolettis and N. D. Fulton. *Software Rejuvenation: Analysis, Module and Ap-*  
1674            *plications*. In Proc. 25th International Symposium on Fault-Tolerant Comput-  
1675            ing, FTCS ’95, pages 381–390. IEEE, 1995. DOI:[10.1109/FTCS.1995.466961](https://doi.org/10.1109/FTCS.1995.466961).
- 1676            [KGG<sup>+</sup>18] P. Kocher, D. Genkin, D. Gruss, W. Haas, M. Hamburg, M. Lipp, S. Mangard,  
1677            T. Prescher, M. Schwarz, and Y. Yarom. *Spectre Attacks: Exploiting*  
1678            *Speculative Execution*. ArXiv e-prints, January 2018.
- 1679            [Kis13] R. Kissel. *Glossary of Key Information Security Terms*. NISTIR 7298  
1680            Revision 2, May 2013. DOI:[10.6028/NIST.IR.7298r2](https://doi.org/10.6028/NIST.IR.7298r2).

- 1681 [KK07] I. Koren and C. M. Krishna. *Fault-Tolerant Systems*. Morgan Kaufmann  
1682 Publishers Inc., 1st edition, 2007.
- 1683 [Koc96] P. C. Kocher. *Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS,*  
1684 *and Other Systems*. In N. Kobitz (ed.), *Advances in Cryptology — CRYPTO*  
1685 *'96*, vol. 1109 of LNCS, pages 104–113. Springer Berlin Heidelberg, 1996.  
1686 DOI:[10.1007/3-540-68697-5\\_9](https://doi.org/10.1007/3-540-68697-5_9).
- 1687 [KPVV16] T. Kaufmann, H. Pelletier, S. Vaudenay, and K. Villegas. *When Constant-Time*  
1688 *Source Yields Variable-Time Binary: Exploiting Curve25519-donna Built*  
1689 *with MSVC 2015*. In *Cryptology and Network Security — CANS 2016*,  
1690 vol. 10052 of LNCS, pages 573–582. Springer Berlin Heidelberg, 2016.  
1691 DOI:[10.1007/978-3-319-48965-0\\_36](https://doi.org/10.1007/978-3-319-48965-0_36).
- 1692 [Kra94] H. Krawczyk. *Secret Sharing Made Short*. In D. R. Stinson (ed.), *Advances*  
1693 *in Cryptology — CRYPTO '93*, vol. 573 of LNCS, pages 136–146. Springer  
1694 Berlin Heidelberg, 1994. DOI:[10.1007/3-540-48329-2\\_12](https://doi.org/10.1007/3-540-48329-2_12).
- 1695 [Lam06] L. Lamport. *Lower bounds for asynchronous consensus*. *Distributed*  
1696 *Computing*, 19(2):104–125, Oct 2006. DOI:[10.1007/s00446-006-0155-x](https://doi.org/10.1007/s00446-006-0155-x).
- 1697 [LS04] B. Littlewood and L. Strigini. *Redundancy and Diversity in Security*.  
1698 In P. Samarati, P. Ryan, D. Gollmann, and R. Molva (eds.), *Computer*  
1699 *Security – ESORICS 2004*, pages 423–438. Springer Berlin Heidelberg, 2004.  
1700 DOI:[10.1007/978-3-540-30108-0\\_26](https://doi.org/10.1007/978-3-540-30108-0_26).
- 1701 [LSG<sup>+</sup>18] M. Lipp, M. Schwarz, D. Gruss, T. Prescher, W. Haas, S. Mangard, P. Kocher,  
1702 D. Genkin, Y. Yarom, and M. Hamburg. *Meltdown*. ArXiv e-prints, January  
1703 2018.
- 1704 [LSP82] L. Lamport, R. Shostak, and M. Pease. *The Byzantine Generals Problem*.  
1705 *ACM Transactions on Programming Languages and Systems*, 4(3):382–401,  
1706 July 1982. DOI:[10.1145/357172.357176](https://doi.org/10.1145/357172.357176).
- 1707 [Lyn89] N. Lynch. *A Hundred Impossibility Proofs for Distributed Computing*. In  
1708 *Proc. 8th Annual ACM Symposium on Principles of Distributed Computing*,  
1709 *PODC '89*, pages 1–28. ACM, 1989. DOI:[10.1145/72981.72982](https://doi.org/10.1145/72981.72982).
- 1710 [MBTM17] K. A. McKay, L. Bassham, M. S. Turan, and N. Mouha. *NISTIR 8114 —*  
1711 *Report on Lightweight Cryptography*, 2017. DOI:[10.6028/NIST.IR.8114](https://doi.org/10.6028/NIST.IR.8114).
- 1712 [MOR01] S. Micali, K. Ohta, and L. Reyzin. *Accountable-subgroup Multisignatures: Extended Abstract*. In *Proc. 8th ACM Conference on Computer*  
1713 *and Communications Security, CCS '01*, pages 245–254. ACM, 2001.  
1714 DOI:[10.1145/501983.502017](https://doi.org/10.1145/501983.502017).
- 1716 [Mor11] T. Morris. *Trusted Platform Module*. In H. C. A. van Tilborg and S. Jajodia  
1717 (eds.), *Encyclopedia of Cryptography and Security*, 2nd Ed., pages 1332–1335.

- 1718 Springer, 2011. DOI:[10.1007/978-1-4419-5906-5\\_796](https://doi.org/10.1007/978-1-4419-5906-5_796).
- 1719 [NIS01] NIST. *Security Requirements for Cryptographic Modules, Federal Information*  
1720 *Processing Standard (FIPS) 140-2*, 2001. DOI:[10.6028/NIST.FIPS.140-2](https://doi.org/10.6028/NIST.FIPS.140-2).
- 1721 [NIS17] NIST. *Post-quantum Cryptography Project*. [https://csrc.nist.gov/projects/](https://csrc.nist.gov/projects/post-quantum-cryptography)  
1722 [post-quantum-cryptography](https://csrc.nist.gov/projects/post-quantum-cryptography), 2017.
- 1723 [NIS18a] NIST. *Automated Cryptographic Validation Protocol*. [https://](https://github.com/usnistgov/ACVP)  
1724 [github.com/usnistgov/ACVP](https://github.com/usnistgov/ACVP), 2018.
- 1725 [NIS18b] NIST. *Automated Cryptographic Validation Testing*. [https://](https://csrc.nist.gov/projects/acvt/)  
1726 [csrc.nist.gov/projects/acvt/](https://csrc.nist.gov/projects/acvt/), 2018.
- 1727 [NIS18c] NIST. *Cryptographic Module Validation Program*. [https://csrc.nist.gov/](https://csrc.nist.gov/projects/cryptographic-module-validation-program)  
1728 [projects/cryptographic-module-validation-program](https://csrc.nist.gov/projects/cryptographic-module-validation-program), 2018.
- 1729 [NRR06] S. Nikova, C. Rechberger, and V. Rijmen. *Threshold Implementations*  
1730 *Against Side-Channel Attacks and Glitches*. In P. Ning, S. Qing, and  
1731 N. Li (eds.), *Information and Communications Security — ICICS 2006*,  
1732 vol. 4307 of LNCS, pages 529–545. Springer Berlin Heidelberg, 2006.  
1733 DOI:[10.1007/11935308\\_38](https://doi.org/10.1007/11935308_38).
- 1734 [NVD14] NVD. *National Vulnerability Database — CVE-2014-0160*.  
1735 <https://nvd.nist.gov/vuln/detail/CVE-2014-0160>, 2014.
- 1736 [NVD18a] NVD. *National Vulnerability Database — CVE-2017-5715*.  
1737 <https://nvd.nist.gov/vuln/detail/CVE-2017-5715>, 2018.
- 1738 [NVD18b] NVD. *National Vulnerability Database — CVE-2017-5753*.  
1739 <https://nvd.nist.gov/vuln/detail/CVE-2017-5753>, 2018.
- 1740 [NVD18c] NVD. *National Vulnerability Database — CVE-2017-5754*.  
1741 <https://nvd.nist.gov/vuln/detail/CVE-2017-5754>, 2018.
- 1742 [OY91] R. Ostrovsky and M. Yung. *How to Withstand Mobile Virus Attacks*  
1743 *(Extended Abstract)*. In Proc. 10th Annual ACM Symposium on Prin-  
1744 ciples of Distributed Computing, PODC '91, pages 51–59. ACM, 1991.  
1745 DOI:[10.1145/112600.112605](https://doi.org/10.1145/112600.112605).
- 1746 [Ped91] T. P. Pedersen. *A Threshold Cryptosystem without a Trusted Party*. In  
1747 D. W. Davies (ed.), *Advances in Cryptology — EUROCRYPT '91*,  
1748 vol. 547 of LNCS, pages 522–526. Springer Berlin Heidelberg, 1991.  
1749 DOI:[10.1007/3-540-46416-6\\_47](https://doi.org/10.1007/3-540-46416-6_47).
- 1750 [Ped92] T. P. Pedersen. *Non-Interactive and Information-Theoretic Secure Verifiable*  
1751 *Secret Sharing*. In J. Feigenbaum (ed.), *Advances in Cryptology — CRYPTO*  
1752 *'91*, vol. 576 of LNCS, pages 129–140. Springer Berlin Heidelberg, 1992.  
1753 DOI:[10.1007/3-540-46766-1\\_9](https://doi.org/10.1007/3-540-46766-1_9).



- 1754 [Por18] T. Pornin. *BearSSL — Constant-Time Mul.* <https://bearssl.org/ctmul.html>,  
1755 2018.
- 1756 [PSL80] M. Pease, R. Shostak, and L. Lamport. *Reaching Agreement in the Presence*  
1757 *of Faults.* J. ACM, 27(2):228–234, April 1980. DOI:[10.1145/322186.322188](https://doi.org/10.1145/322186.322188).
- 1758 [PW92] B. Pfitzmann and M. Waidner. *Unconditional Byzantine agreement for any*  
1759 *number of faulty processors.* In A. Finkel and M. Jantzen (eds.), STACS 92,  
1760 pages 337–350. Springer Berlin Heidelberg, 1992. DOI:[10.1007/3-540-55210-](https://doi.org/10.1007/3-540-55210-3_195)  
1761 [3\\_195](https://doi.org/10.1007/3-540-55210-3_195).
- 1762 [Rab83] M. O. Rabin. *Randomized Byzantine Generals.* In Proc. 24th Annual  
1763 Symposium on Foundations of Computer Science, SFCS '83, pages 403–409.  
1764 IEEE Computer Society, 1983. DOI:[10.1109/SFCS.1983.48](https://doi.org/10.1109/SFCS.1983.48).
- 1765 [Rad97] J. Radatz. *The IEEE Standard Dictionary of Electrical and Electronics Terms.*  
1766 IEEE Standards Office, 6th edition, 1997.
- 1767 [RSA78] R. L. Rivest, A. Shamir, and L. Adleman. *A method for obtaining digital*  
1768 *signatures and public-key cryptosystems.* Communications of the ACM,  
1769 21(2):120–126, 1978. DOI:[10.1145/359340.359342](https://doi.org/10.1145/359340.359342).
- 1770 [RSWO17] E. Ronen., A. Shamir, A.-O. Weingarten, and C. O'Flynn. *IoT Goes Nuclear:*  
1771 *Creating a ZigBee Chain Reaction.* IEEE Symposium on Security and Privacy,  
1772 pages 195–212, 2017. DOI:[10.1109/SP.2017.14](https://doi.org/10.1109/SP.2017.14).
- 1773 [SA09] N. R. Sunitha and B. B. Amberker. *Forward-Secure Multi-signatures.*  
1774 In M. Parashar and S. K. Aggarwal (eds.), Distributed Computing and  
1775 Internet Technology, pages 89–99. Springer Berlin Heidelberg, 2009.  
1776 DOI:[10.1007/978-3-540-89737-8\\_9](https://doi.org/10.1007/978-3-540-89737-8_9).
- 1777 [Sau34] R. Saunders. *Poor Richard's Almanack — 1735.* Benjamin Franklin, 1734.
- 1778 [Sch90] C. P. Schnorr. *Efficient Identification and Signatures for Smart Cards.*  
1779 In G. Brassard (ed.), Advances in Cryptology — CRYPTO'89 Proceed-  
1780 ings, vol. 435 of LNCS, pages 239–252. Springer New York, 1990.  
1781 DOI:[10.1007/0-387-34805-0\\_22](https://doi.org/10.1007/0-387-34805-0_22).
- 1782 [Sch99] B. Schoenmakers. *A Simple Publicly Verifiable Secret Sharing Scheme*  
1783 *and Its Application to Electronic Voting.* In M. Wiener (ed.), Advances in  
1784 Cryptology — CRYPTO '99, vol. 1666 of LNCS, pages 148–164. Springer  
1785 Berlin Heidelberg, 1999. DOI:[10.1007/3-540-48405-1\\_10](https://doi.org/10.1007/3-540-48405-1_10).
- 1786 [Sha97] W. Shakespeare. *An excellent conceited Tragedie of Romeo and Juliet.* Printed  
1787 by John Danter, London, 1597.
- 1788 [Sha79] A. Shamir. *How to Share a Secret.* Communications of the ACM,  
1789 22(11):612–613, Nov 1979. DOI:[10.1145/359168.359176](https://doi.org/10.1145/359168.359176).

- 1790 [Sho00] V. Shoup. *Practical Threshold Signatures*. In B. Preneel (ed.), *Advances*  
1791 *in Cryptology — EUROCRYPT 2000*, vol. 1807 of LNCS, pages 207–220.  
1792 Springer Berlin Heidelberg, 2000. DOI:[10.1007/3-540-45539-6\\_15](https://doi.org/10.1007/3-540-45539-6_15).
- 1793 [SNV07] P. Sousa, N. F. Neves, and P. Verissimo. *Hidden Problems of Asynchronous*  
1794 *Proactive Recovery*. Proc. 3rd Workshop on on Hot Topics in System  
1795 Dependability, 2007.
- 1796 [Sta96] M. Stadler. *Publicly Verifiable Secret Sharing*. In U. Maurer (ed.), *Advances*  
1797 *in Cryptology — EUROCRYPT ’96*, vol. 1070 of LNCS, pages 190–199.  
1798 Springer Berlin Heidelberg, 1996. DOI:[10.1007/3-540-68339-9\\_17](https://doi.org/10.1007/3-540-68339-9_17).
- 1799 [tC96] U. S. 104th Congress. *Information Technology Management*  
1800 *Reform Act. Public Law 104–106, Section 5131*, 1996. [https:](https://www.dol.gov/ocfo/media/regs/ITMRA.pdf)  
1801 [://www.dol.gov/ocfo/media/regs/ITMRA.pdf](https://www.dol.gov/ocfo/media/regs/ITMRA.pdf).
- 1802 [TJ11] H. C. A. Tilborg and S. Jajodia. *Encyclopedia of Cryptography and*  
1803 *Security*. Springer Publishing Company, Incorporated, 2nd edition, 2011.  
1804 DOI:[10.1007/978-1-4419-5906-5](https://doi.org/10.1007/978-1-4419-5906-5).
- 1805 [Vas15] A. Vassilev. *Cryptographic Validation Challenges With Brittle Algo-*  
1806 *rithms*. [https://csrc.nist.gov/groups/ST/lwc-workshop2015/presentations/](https://csrc.nist.gov/groups/ST/lwc-workshop2015/presentations/session5-vassilev.pdf)  
1807 [session5-vassilev.pdf](https://csrc.nist.gov/groups/ST/lwc-workshop2015/presentations/session5-vassilev.pdf), July 2015.
- 1808 [VCB<sup>+</sup>13] G. S. Veronese, M. Correia, A. N. Bessani, L. C. Lung, and P. Verissimo.  
1809 *Efficient Byzantine Fault-Tolerance*. IEEE Transactions on Computers,  
1810 62(1):16–30, 01 2013. DOI:[10.1109/TC.2011.221](https://doi.org/10.1109/TC.2011.221).
- 1811 [vdGP88] J. van de Graaf and R. Peralta. *A Simple and Secure Way to Show the Validity*  
1812 *of Your Public Key*. In C. Pomerance (ed.), *Advances in Cryptology —*  
1813 *CRYPTO ’87*, vol. 293 of LNCS, pages 128–134. Springer Berlin Heidelberg,  
1814 1988. DOI:[10.1007/3-540-48184-2\\_9](https://doi.org/10.1007/3-540-48184-2_9).
- 1815 [Ver18] Verizon. *2018 Data Breach Investigations Report*. [https:](https://www.verizonenterprise.com/verizon-insights-lab/dbir/)  
1816 [://www.verizonenterprise.com/verizon-insights-lab/dbir/](https://www.verizonenterprise.com/verizon-insights-lab/dbir/), 2018.
- 1817 [VMB18] A. Vassilev, N. Mouha, and L. Brandão. *Psst, Can You Keep a Secret?* IEEE  
1818 Computer, 51(1):94–97, January 2018. DOI:[10.1109/MC.2018.1151029](https://doi.org/10.1109/MC.2018.1151029).
- 1819 [Yao82] A. C. Yao. *Protocols for secure computations*. In 23rd Annual Symposium  
1820 *on Foundations of Computer Science, SFCS ’82*, pages 160–164, 11 1982.  
1821 DOI:[10.1109/SFCS.1982.88](https://doi.org/10.1109/SFCS.1982.88).
- 1822 [Yao86] A. C. Yao. *How to Generate and Exchange Secrets*. In Proc. 27th Annual  
1823 *Symposium on Foundations of Computer Science, SFCS ’86*, pages 162–167.  
1824 IEEE Computer Society, 1986. DOI:[10.1109/SFCS.1986.25](https://doi.org/10.1109/SFCS.1986.25).