Threshold Schemes for Cryptographic Primitives

Challenges and Opportunities in Standardization and Validation of Threshold Cryptography

Luís T. A. N. Brandão
Nicky Mouha
Apostol Vassilev

This publication is available free of charge from:
https://doi.org/10.6028/NIST.IR.8214
Threshold Schemes for Cryptographic Primitives

Challenges and Opportunities in Standardization and Validation of Threshold Cryptography

Luís T. A. N. Brandão
Nicky Mouha
Apostol Vassilev

Computer Security Division
Information Technology Laboratory

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

July 2018 - March 2019

U.S. Department of Commerce
Wilbur L. Ross, Jr., Secretary

National Institute of Standards and Technology
Walter G. Copan, NIST Director and Under Secretary of Commerce for Standards and Technology
This is a diff, between the Draft (July 2018) and Final (March 2019) versions. Review comments are included in the end.

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.

Public-comment-period Comments on this publication may be submitted to: 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

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

The Computer Security Division at the National Institute of Standards and Technology is interested in promoting the security of implementations of cryptographic primitives. This security depends not only on the theoretical properties of the primitives but also on the ability to withstand attacks on their implementations. It is thus important to mitigate breakdowns that result from differences between ideal and real implementations of cryptographic algorithms.

This document overviews threshold cryptographic schemes, which enable attaining the possibility of implementing cryptographic primitives using threshold schemes, where multiple components contribute to the operation in a way that attains the desired security goals even if \( f \) out of \( n \) of its components are compromised. There is also an identified potential in providing resistance against side-channel attacks, which exploit inadvertent leakage from real implementations. Security goals of interest include the secrecy of cryptographic keys, as well as enhanced integrity and availability, among others.

This document considers challenges and opportunities related to standardization of threshold schemes for cryptographic primitives. It includes examples illustrating security tradeoffs under variations of system model and adversaries. It enumerates several high-level characterizing features of threshold schemes, including the types of threshold, the communication interfaces (with the environment and between components), the executing platform (e.g., single device vs. multiple devices) and the setup and maintenance requirements.

The document poses a number of questions, motivating aspects to take into account when considering standardization. A particular challenge is the development of criteria that may help guide a selection of threshold cryptographic schemes. An open question is deciding at what level each standard should be defined (e.g., specific base techniques vs. conceptualized functionalities) and which flexibility of parametrization they should allow. Suitability to testing and validation of implementations are also major concerns to be addressed. Overall, the document intends to support discussion about standardization, including motivating an engagement from stakeholders. This is a step towards enabling threshold cryptography within the US federal government and beyond.

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

Acknowledgments

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

The initial draft of this report was published online on July 26, 2018, for a period of public comments. We have since then received valuable feedback from external reviewers, which allowed us to improve the quality of the final document. We thank the comments received (in chronological order) from the following persons: Svetla Nikova from KU Leuven; Gokhan Kocak from Asena Inc.; Oliver Stengele from the Karlsruhe Institute of Technology; Aivo Kalu from Cybernetica AS; Christian Cachin, Hugo Krawczyk, Tal Rabin, Jason Resch and Chrysa Stathatkopoulou from IBM Research; Tanja Lange from Technische Universität Eindhoven; Yehuda Lindel from Bar-Ilan University and Unbound Tech; Samuel Ranellucci from Unbound Tech; Dan Bogdanov from Cybernetica AS; Rosario Gennaro from The City College of New York; Thalia May Laing from HP Labs; Karim Eldefrawy from SRI International; John Wallrabenstein from Analog.
Executive Summary

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

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

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

In threshold cryptography, the shares of the key do not need to be recombined to compute a particular result. Instead, the parties independently or collaboratively calculate shares of the output, without revealing the input shares to one another. This may be facilitated by certain mathematical properties, such as homomorphisms, or by cryptographic “secure computation” protocols. Using the threshold property, the output from the share computation can then be reconstructed into a final output. This is possible to achieve for NIST-approved algorithms, such as RSA and DSA—Rivest–Shamir–Adleman (RSA) and Digital Signature Algorithm (DSA) signatures, and AES—Advanced Encryption Standard (AES) enciphering and deciphering.

Threshold schemes can be used, with different security goals, in different applications. For example: (i) implement a digital signature algorithm without any single component ever holding the signing key; (ii) implement encryption and decryption correctly even if one compromised component attempts to corrupt the output; (iii) generate unbiased randomness.
even if some (but not all) randomness contributors are biased or unavailable.

The computational paradigm in threshold cryptography brings several security advantages but also some potential weaknesses. For example, the use of multiple shares increases the attack surface to encompass all shares. Thus, the security effect of implementing a threshold scheme depends on an attack model. It is particularly relevant to consider how difficult it may be to compromise more than the threshold number of components. In some cases, for example with low $f$, the increased attack surface may enable an attack more efficient and effective than possible against a conventional (non-threshold) primitive—*even if the nodes in the threshold scheme have independent modes of compromise* (e.g., each compromisable via mutually exclusive attack vectors).

On the other hand, a threshold scheme may provide better security even if the components are individually easier to compromise, e.g., in some settings/models where they are also easier to patch.

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

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

Separating the analysis of different security aspects can sometimes lead to pitfalls. To avoid such problems it is important to use appropriate formal models of security. At the same time, it is relevant to assess potential tradeoffs that a threshold cryptographic scheme induces across different security properties. A system model is also important to characterize different types of attack that a system may be subject to. Specific attacks in the real world exploit differences between conventional implementations and their idealized versions.

Threshold schemes can be used to improve resistance against some of these specific attacks that breach specific security properties (e.g., confidentiality of a key) or sets thereof.

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

The examples in the document illustrate how security properties can vary depending on high-level features, on assumed attack vectors and on the type of adversarial goals and capabilities. On one hand, this helps prevent a possible misconception that a higher threshold directly means higher security. On the other hand, it also intends to convey that threshold schemes can be used to implement cryptographic primitives in a more secure way. Altogether, structured security assertions also promote a path for meaningful security validation of actual implementations.

This document considers the benefits of standardizing threshold cryptographic schemes, possibly along with auxiliary threshold-cryptography primitives. Naturally, there is interest on threshold schemes for NIST-approved cryptographic primitives. Also of major importance is the development of corresponding approaches for validation of implementations of threshold cryptographic schemes. This should be aligned with the current modernization process and evolving structure of the testing methodology of the NIST cryptographic validation programs. Of particular relevance is the development of approaches to enable automated validation tests with state-of-the-art techniques.

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

There are important challenges and opportunities related to the standardization of threshold cryptographic schemes. Addressing these may bring about important security improvements to real implementations of cryptographic primitives. Fortunately, there is a plethora of research work done in the broad area of threshold cryptography, providing useful insights about possible options, caveats and tradeoffs. Further value can arise from addressing these challenges with feedback and collaboration from stakeholders, including academic researchers, industry participants and government representatives.
This publication is available free of charge from: https://doi.org/10.6028/NIST.IR.8214

NISTIR 8214 (Draft)

This is a diff, between the Draft (July 2018) and Final (March 2019) versions. Review comments are included in the end.

Table of Contents

1 Introduction 1

2 Fundamentals 4

2.1 Terminology . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4

2.2 Secret sharing . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6

2.3 Secret resharing . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 8

2.4 Threshold cryptography . . . . . . . . . . . . . . . . . . . . . . . . . . . 9

2.5 Side-channel and fault attacks . . . . . . . . . . . . . . . . . . . . . . . . 11

3 Examples 12

3.1 Threshold signature examples . . . . . . . . . . . . . . . . . . . . . . . . . 12

3.2 Examples of side-channel attacks and countermeasures . . . . . . . 15

4 Models 16

4.1 Security considerations . . . . . . . . . . . . . . . . . . . . . . . . . . . . 17

4.2 Types of attack . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 21

4.3 System model . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 24

5 Characterizing features 28

5.1 Threshold values . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 28

5.2 Communication interfaces . . . . . . . . . . . . . . . . . . . . . . . . . . 31

5.3 Target computing platforms . . . . . . . . . . . . . . . . . . . . . . . . . . 33

5.4 Setup and maintenance . . . . . . . . . . . . . . . . . . . . . . . . . . . . 37

6 Validation of implementations 40

6.1 The existing CMVP and FIPS 140-2 . . . . . . . . . . . . . . . . . . . . . 40

6.2 Integration of threshold cryptographic schemes . . . . . . . . . . . . . . 41

7 Criteria for standardization 42

7.1 Representative questions . . . . . . . . . . . . . . . . . . . . . . . . . . . 42

7.2 Standardization at what granularity level? . . . . . . . . . . . . . . . . . 44

7.3 Standardization opportunities . . . . . . . . . . . . . . . . . . . . . . . . 45

8 Conclusions 46

References 48

List of Figures

1 Illustration of Blakley secret sharing . . . . . . . . . . . . . . . . . . . . . 7

2 Illustration of share randomization in Blakley secret sharing . . . . . . . 9

List of Tables

1 Representative attack types . . . . . . . . . . . . . . . . . . . . . . . . . . 22
1 Introduction

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

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

To counter the inherent security risks of handling secret keys in conventional implementations of cryptographic algorithms, technical approaches have emerged that split the secret key into two or more shares across different components or parties. For example, upon using secret sharing the compromise of one (or more, but not all when secret sharing is used on the key, the compromise of up to the confidentiality threshold number \( f \) (out of \( n \)) of the shares does not reveal information about the original key. Using appropriate threshold techniques, the shares can then be separately processed, leading the computation to a correct result as if the original secret key had been processed by a classic algorithm. The threshold approach can thus significantly increase the confidentiality of secret keys in cryptographic implementations.

¹ Some portions of writing were adapted from text appearing at a previous short magazine article [VMB18].
issues in hardware and software implementations. The threshold approach can also enable
decentralization of trust, when delegating the ability to perform some cryptographic operation.
This can be useful for higher-level distributed applications, e.g., when the performing of a
cryptographic operation should require agreement by multiple parties.

This report is focused on threshold schemes applied to cryptographic primitives. In
an \( f \)-out-of-\( n \) threshold scheme, some security property is tolerant to \textbf{the same kind}
of compromise of up to \( f \) out of \( n \) components in the system. \textbf{This - As a mnemonic,}
the symbol \( f \) can be thought of as counting the number of “faulty” (i.e., compromised)
components that can be tolerated. \textbf{This threshold \( f \) can be specific to some implicit type}
of compromise, e.g., possibly including cases of crash, leakage, intrusion and accidental or
malicious malfunctioning.

In a dual perspective, a threshold can be defined with respect to an operational property.
A \( k \)-out-of-\( n \) threshold property denotes that the presence or participation of \( k \) correct
components is required to ensure some correct operation. The relation between the different
thresholds (respectively represented by symbols \( k \) and \( f \)), e.g., \( k = f + 1 \) or \( k = 2f + 1 \), can
vary depending on the scheme and on the type of compromise and security property.

The threshold paradigm brings several security advantages but also some potential weak-
nesses. For example, the use of multiple shares increases the attack surface to encompass
all shares. Thus, the security effect of implementing a threshold scheme depends on an
attack model. It is particularly relevant to consider how difficult may be the compromise of
more than the threshold number \( f \) of components. In some cases, for example with low \( f \),
the increased attack surface may enable an attack more efficient and effective than possible
against a conventional (non-threshold) primitive.

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

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

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

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

There are techniques designed to mitigate foreseen compromises in more complicated scenarios. For example, verifiable secret-sharing enables detection of misuse of shares by a shareholder, thereby enabling operational modes that tolerate this kind of corruption. As another example, proactive secret sharing can be used to periodically reshare a secret, thereby periodically reducing to zero the number of compromised shares. However, an abstract security model is not enough to assess the effects of specific attacks. It is thus important to develop objective criteria for selecting from a potential pool of candidate threshold schemes.

Altogether, the security assertions made with respect to an instantiated set of features provide a path for security validation of actual implementations. Of particular interest are approaches that enable automated validation tests with state-of-the-art techniques. The use of well-characterized threshold cryptographic schemes to implement cryptographic primitives offers potential security benefits. It is thus important to develop objective criteria for selecting from a potential pool of candidate threshold schemes.
Audience. This document is targeted, with varying goals, at a diverse audience. Internally for NIST, the goal is to initiate a discussion about threshold schemes for cryptographic primitives. This motivated the inclusion of representative questions relevant to standardization.

The document is also written for people with managerial/policy responsibilities in development and/or adoption of cryptographic services and modules. For such an audience, the document highlights critical aspects of the security of implementations that can be significantly affected by nuances in the system model and the employed threshold techniques. Several simple examples are provided, including some based on classic secret sharing schemes.

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

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

2 Fundamentals

[[The subsection “Terminology” was previously the last subsection (2.5) of Section 2]]

2.1 Terminology

This document makes use of two dual perspectives of a threshold. In “f-out-of-n” the threshold “f” denotes the maximum number of components that can be compromised (with respect to some implicit security property of the components), while retaining some (implicit) security property for the global system. Correspondingly, in “k-out-of-n” the threshold “k” denotes the minimum number of components that must remain uncompromised to be possible to ensure some security property of the global system.

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

- **Active/byzantine/malicious**: characterization of compromised nodes, or of an adversary, when being able to arbitrarily deviate or induce deviations from a protocol specification.

- **Agent/component/node/party/share**: a constituent part of an implemented threshold scheme, affecting the prosecution of a functional goal (a cryptographic operation, in our context) to be achieved by a collective of parts; most often used to denote one of the $n$ parts whose compromise counts towards the threshold $f$; when the context is clear, some terms can designate parts outside of the threshold composition.

- **Aggregator/broker/combiner/dealer/proxy/relay**: an agent with a special role in aiding the setup, execution and/or maintenance of a threshold protocol; usually not accounted in $n$, except if explicitly stated as such (e.g., the case of a primary node).

- **Bad/compromised/corrupted/faulty/intruded**: state of a node, whereby it departs from an ideally healthy state, and starts being counted towards the threshold $f$.

- **Client/user**: an agent, not in the threshold set of components, who is a stakeholder of the result of a cryptographic computation, typically the requester for that computation.

- **Compromise/corruption/intrusion**: a process by which a node transitions from an ideally healthy state to a compromised state and/or by which it remains therein.

- **Good/healthy/honest/recovered**: ideal state of a node, not yet compromised by an adversary, but susceptible to attacks.

- **Honest-but-curious/Leaky/Passive/Semi-honest**: characterization of compromised components, or of an adversary, when the internal state of the former is exfiltrated by the latter, but without altering the computations and message-exchanges specified by the protocol.

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

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

Secret sharing is based on splitting the key into multiple shares. For example, to split key $K$ into three shares $K_1$, $K_2$, and $K_3$, we randomly select shares $K_1$ and $K_2$ from the same key space as $K$, and let the third share $K_3 = K_1 \oplus K_2 \oplus K$ be the one-time pad encryption of $K$, where $\oplus$ is the exclusive OR operation if the keys are bit-strings. No two shares provide any information about the secret key — all shares are required to recover $K$.

The described scheme has a “3-out-of-3” property, with $n = 3$ parties, has a threshold property: it is a “f-out-of-3” scheme with $f = 2$ with respect to the leakage of any two shares alone not giving away information of the original secret key; it is a $k$-out-of-$3$ scheme with $k = 3$ with respect to all three shares being required to recover the key. The $k$ notation is used hereafter for the concrete case of secret-sharing schemes.

More generally, $k$-out-of-$n$ secret-sharing schemes can be defined, for any integers $n$ and $k$ satisfying $n \geq k \geq 1$. Such secret-sharing schemes were independently developed in 1979 by Shamir [Sha79] and Blakley [Bla79]. There, any $k$ parties together can recover a secret shared across $n$ parties, but $k - 1$ parties together do not know anything about the secret.

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

Similarly, if Bob and Charlie obtain coefficients of other lines that pass through the same point $P$, individually they cannot determine $P$. Note that the lines cannot be parallel to each other and to Alice’s line. However, any two together — Alice with Bob, or Alice with Charlie, or Bob with Charlie — can easily compute $P$ as the intersection of their lines (see Fig. 1(c)). We have thus described a 2-out-of-3 secret-sharing scheme. To build a $k$-out-of-$n$ Blakley scheme for some $k > 2$, one considers hyperplanes $y = h_1x_1 + \ldots + h_kx_k + g$ that intersect in a single point $P(x_1, \ldots, x_{k-1})$ in the $k$-dimensional space. The coefficients $h_r$ are non-zero and $g$ is an arbitrary constant, provided that no hyper-place is orthogonal to the $x_1$-axis. Choosing $n \geq k$ such hyperplanes, one can distribute the corresponding coefficients to $n$ different parties. Then any $k$ parties together can compute efficiently the intersection point $P$. The prime modulus $Q$ must be larger than the secret $x_s$ and larger than
the number $n$ of parties and recover the secret as its $x_1$-coordinate.

The number $n$ of parties and recover the secret as its $x_1$-coordinate.

The schemes of Shamir and Blakley are information-theoretically secure, which means that indeed there is no information about the key in a standalone set of $k \leq$ shares. This means that the scheme can in practice be used to share very small secrets (e.g., only a few bits), independently of the application. If, however, the sharing is applied to a cryptographic key required to be larger than some security parameter, e.g., 256 bits, then the corresponding prime $Q$ must be correspondingly large. Alternatively, the secret sharing could be applied in parallel to independently share portions of a secret.

While information-theoretic security may be an advantage, the property requires that each share be independent.

---

1 The humanoid cliparts are from clker.com/clipart-*.html, where * is 2478, 2482, and 2479.

2 The humanoid cliparts are from clker.com/clipart-*.html, where * is 2478, 2482, and 2479.
506 2.3 Secret resharing

The need to compute new random shares for the same original secret key often arises in practice. It may happen that over time some (< k) shares are compromised [OY91], thus creating a need to compute new shares and discard the old ones. Resharing can even be proactive [HJKY95], e.g., at regular intervals in time and not as a direct response to a detected compromise.

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

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

For each new iteration r + 1, one computes a new point \(P_{r+1} = (x_s, y_{r+1})\) and new random
lines for each party. These lines, passing through point $P_{r+1}$ correspond to new random angles, as illustrated in Fig. 2(b). The dealer (i.e., the party selecting new shares) must ensure that the lines of different parties do not overlap, i.e., that they do not have the same angles, and are non-vertical. Concretely, this means that $\omega_{i,r} \neq \omega_{j,r} \neq \pi/2$ for $i, j \in \{A, B, C\}$ and $i \neq j$. The generalization to the case $k > 2$ is as before: the new point $P$ would require randomizing $k - 1$ coordinates, and the resharing would proceed as in the initial sharing.

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

Note: several elements of secret-sharing are standardized by ISO/IEC.

2.4 Threshold cryptography

We take broad input from several research areas with traditionally distinctive names, but with a strong relation to threshold schemes. Since we are focused on the implementation of cryptographic primitives, we adopt the umbrella term “threshold cryptography” to denote our area of interest. The expression “threshold cryptography” has been traditionally used to refer to schemes where some computation is performed over secret shares of inputs [DF90, DSDFY94]. Usually, the setting is such that the shares are used to compute something useful, but without being revealed across parties. Often, a main security goal is
secrecy of cryptographic keys, but a variety of other security properties, such as integrity and availability, may also be a motivating drive. Achieving these properties is possible based on a variety of techniques. For example, integrity may in some settings be enhanced based on verifiable secret sharing schemes [AMGC85, Fel87] and/or zero-knowledge proofs [GMR85, BFM88], allowing checking whether shares are used consistently. Specifically, a threshold scheme can be made robust against adversarially induced inconsistencies in shares or in related computations, outputting correct results in spite of up to a threshold number of compromised parties [GRJ00]. While we focus on secure implementations of cryptographic primitives, the actual threshold techniques may also include non-cryptographic techniques, e.g., simple replication and majority voting.

Threshold schemes also do not encompass all that exists in the realm of SMPC. In usual SMPC descriptions, the parties themselves are stakeholders of the secrecy of their input and correctness of the output, e.g., in the millionaire’s millionaires’ problem [Yao82] and in secure set intersection [FNP04]. Conversely, threshold schemes are often envisioned at a higher level, where the threshold entity has In threshold schemes for cryptographic primitives, the nodes within the threshold system can have a neutral interest for the outcome, and in fact just be a service provider (of cryptographic services) to a set of external users/clients. In other words, threshold schemes do not encompass all that exists in the realm of SMPC, and vice versa there are threshold schemes not based on SMPC.

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

2.5 Side-channel and fault attacks

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

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

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

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

3.1 Threshold signature examples

3.1.1 Basic threshold computation on secret shares

Now let us proceed to construct a threshold scheme for digital signatures. First, we recall the RSA (Rivest-Shamir-Aleman) signature scheme [RSA78], which defines the public key as \((N, e)\) and the private key as \(d\), such that \(ed = 1 \mod \phi(N)\). Here, the modulus \(N\) is a product of two large secret primes and \(\phi\) is Euler’s totient function. Then, the RSA signature for a (possibly hashed) message \(m\) is defined as \(s = m^d \mod N\). Anyone possessing the public key can verify the signature by checking \(s^e = m^d = m \mod N\). To obtain a-

Now let us proceed to describe a simple threshold variant of this signature scheme. We split the [Fra90, BH98]. We first consider the role of a dealer — someone that, knowing the secret parameter \(\phi(N)\) and the secret signing key \(d\), wishes to delegate to other parties the ability to jointly produce signatures in a threshold manner. The dealer splits the private key \(d\) into three shares \(d_1, d_2,\) and \(d_3\), such that \(d_1 + d_2 + d_3 = d \mod \phi(N)\). Now, without reconstructing \(d\), it is possible to first process 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 then compute the signature \(s = s_1s_2s_3\). Note that this is indeed a valid RSA signature, as \(s_1s_2s_3 = m^{d_1+d_2+d_3} = m^d \mod N\). This simple threshold RSA signature scheme mitigates the risk of exposing the potentially high-value private key \(d\), which does not appear in any of the three shares that are used in the actual computations. Thus, compromising any one of the shares, and even two of them, poses no threat of exposing \(d\). Moreover, frequent updates to the key shares \((d_1, d_2,\) and \(d_3)\) would reduce the window of opportunity for attacks and thereby further reduce the risk. Refreshing can even occur after every signature.

3.1.2 A \(k\)-out-of-\(n\) threshold scheme

In the above example, all shares must be present. This might be impractical in situations where one or more of the shares become unavailable. For such cases, a \(k\)-out-of-\(n\) threshold scheme could be used when at least \(k\) shares are available. For RSA signatures, —

But how to generalize from the \(n\)-out-of-\(n\) to a \(k\)-out-of-\(n\) signature scheme (with \(k < n\))? The needed secret-sharing is no longer a simple additive decomposition into shares (in the exponent), and correspondingly the combination into a final signature becomes more
complex, namely because the share-holders do not know \( \phi(N) \) (the group order). It is nonetheless possible to slightly adjust the computation of key shares and signature shares so that the final combination becomes possible even without knowledge of the secret information [Sho00]. The secret vs. public knowledge of the order of the underlying group can indeed be relevant in the development of diverse threshold schemes, with some schemes taking advantage of using groups with publicly known group order (e.g., as in the case of ElGamal decryption [DF90]).

So for RSA signatures one can use a 2-out-of-3, e.g., a 2-out-of-3 secret-sharing scheme, and a corresponding threshold variant of RSA. Then, in the case of one share being irrecoverably lost or breached, the private signature key \( d \) remains intact, available, and not breached. This means that one can continue to use the same public key to verify the signature’s correctness—correctness of the signature.

In contrast, when a conventional implementation is breached, the corresponding public/private key pair would have to be revoked and a new pair issued. Typically this also requires an external certification of the public key by a certificate authority and propagating it to all relying parties. In addition, a 2-out-of-3 threshold signature scheme becomes more resilient to future share losses if it continuously refreshes the key shares, provided that at most one is compromised at any given time. Note that in a scheme composed of three separate conventional RSA implementations with independent keys, refreshing would require updating the public/private key pairs, along with all entailing inconveniences.

Avoiding the dealer.

### 3.1.3 Avoiding the dealer

In the above descriptions, an implicit trusted party, often called the dealer, knows the secret \( d \) and performs each secret-sharing operation. Particularly, the threshold RSA examples based on a common modulus \( N \) required the dealer to also know the secret prime factorization. Without knowledge of such factorization, it is not currently known how to correctly select such modulus and prove its correctness. Thus, the selection of secrets If needed, the dealer could, without revealing the secret, prove to each share-holder that \( N \) is indeed a valid product of two primes, by using zero-knowledge proofs [vdGP88]. Nonetheless, in a dealerless setting a threshold computation of shares of the secret signing key \( d \) would also require a threshold generation of the public \( N \), along with a secret sharing of its factorization. This does not lend itself to a straightforward efficient threshold computation. Nonetheless, such threshold selection, even for RSA keys, can still be done based on SMPC protocols [BF97]. By using zero-knowledge proofs, the needed property on \( N \) can also be proven without revealing the secret. Different RSA-based threshold schemes can take advantage of specialized solutions, with tradeoffs related to the threshold parameters \( k \) and \( n \) and to properties of the prime factorization.
Schemes based on different assumptions can enable a more straightforward selection and verification of the validity of public elements. For example, this is possible based on assumptions of intractability of computing discrete logarithms in certain groups of known order. If the group parameters can be verified as correct in a standalone procedure, then no one requires knowing any secret knowledge about the group. Furthermore, if the selection is made in a way that avoids the possibility of a trapdoor being known, then the parameters can be trusted by anyone. The intractability assumption can then, for fixed security parameters, be accepted for universal parameters of a group (e.g., [Ber06]).

In particular, this can facilitate a respective threshold mechanism, so that a secret key never exists locally at any entity. For example, one can then define a dealer-absent threshold version of a public key generation (the result of an exponentiation), such that each party knows one share of the secret key (a discrete logarithm) [Ped91, GJKR99].

**Other constructions.** The same possibilities exist for resharings. In suitable threshold schemes, the share-holders can perform resharings without a dealer, i.e., interacting to create new shares for the same secret, without ever reconstructing the secret. The final shares can even be obtained for new threshold structures (e.g., different threshold and number of parties) [DJ97].

**3.1.4 Other constructions**

The above examples focused on threshold schemes where the secret-key is shared, and then a threshold scheme enables a generation of a signature identical to the non-threshold manner. A feature of those schemes is that the final signature is identical to a non-threshold one, thereby being inherently efficient in size (i.e., not incurring an increase with the threshold parameter). Such schemes also have the property that the identities of the signatories remain secret to the external users interested in verifying the correctness of a signature. However, some settings may favor the identifiability of signatories, e.g., as an accountability and credibility feature. Each signatory might also prefer retaining an individual public credential, not wanting to use a private-key share associated with a common public key. Even in this setting it is possible to devise short threshold signatures, with size equal to a non-threshold signature. Concretely, “multi-signature” schemes [IN83, MOR01] enable multiple parties, with independent secret-public key pairs, to jointly produce a common short signature.3

A multi-signature scheme can be used as a threshold signature scheme where the application layer, and possibly the user, has added flexibility to define which subsets of signatories determine a valid signature, i.e., beyond structures defined by a pre-determined threshold number. For example, a multi-signature may be defined as valid if it contains one signature from each of three groups of individuals in different roles in an organization. The

---

3 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.
verification procedure then depends on the set of independent public keys. For example, these schemes can be easily based on Schnorr signatures [Sch90, BN06].

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

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

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

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

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

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

Even worse, may not be sufficient. Indeed, an implementation that is free of timing attacks on one platform may be vulnerable on another platform. This can happen, for example, when source code that contains multiplication operations is compiled with a different runtime library [KPVV16], or when the same binary is executed on a different processor [Por18].

The execution time of the program, however, is just one example of a side channel. Implementations in hardware and software may also leak through other side channels, such as power consumption or electromagnetic radiation. The limitation of the currently-known countermeasures (such as “constant-time” implementations, dual-rail logic, or electromagnetic shielding) is that they usually do not get rid of all the leakage, but may still be vulnerable to higher-order or data-dependent leakages.
To protect against side-channel attacks, the framework of threshold cryptography can provide a promising starting point. If the implementation is split into a number of “parties,” such that no single party holds the entire secret required to perform the cryptographic operation, then the leakage of information from only one “party” would not enable a successful attack on the original secret.

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

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

### 4 Models

The basic security model for conventional cryptographic algorithms assumes an ideal black box, in which the cryptographic computations are correct and all internal states, including keys, are kept secret. Such ideal constructs would not leak any secret information. This basic model leaves aside the possibility of leakage through side-channels, such as timing and power. In other words, in the ideal black box the time and energy used for operations would be independent of secrets. This may be due to these parameters not being included in the model, or being assumed independent of the secrets (e.g., being instantaneous or requiring constant time assuming instantaneous or constant-time computations). Under this assumption, one can reduce the problem of evaluating the algorithm’s security properties to the quantifying a security property of the algorithm can be reduced to the problem of evaluating the complexity of the best-known attack against this model.

For example, one can define the security strength, which can also be expressed as bit strength, of different classes of cryptographic algorithms based on the amount of work needed to perform a brute-force search of the key in a large space related to the key size. When the algorithms are implemented in real hardware and software, the black-box assumption can break down in several ways. For example, bugs in the implementation can lead to side effects that compromise the secret key, as with Heartbleed. Also, the material and electromagnetic characteristics of the platforms on which the algorithms run can cause side-channel information to leak and allow attackers to recover the secret key.

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

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

4.1 Security considerations

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

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

4.1.1 Diverse security properties
A threshold augmentation may have different effects across different security properties, e.g., confidentiality vs. availability vs. integrity, possibly improving one while degrading others. To show the nuances, consider the threshold 3-out-of-3 threshold RSA-signature scheme described in Sec. 3.1.1, supported on a 3-out-of-3 secret sharing of the key. (Recall that, with the notation used here, a 3-out-of-3 secret sharing of the key means \( k = 3 \) for availability, i.e., three parties (out of \( n = 3 \)) are necessary to produce a signature, and \( f = 2 \) for confidentiality, i.e., any subset of only two parties cannot learn anything about the key.)

There, each node loses visibility of the original signing key, but retains the ability to influence the output of a computation dependent on the key. If a compromised node simply refrains from outputting, then it compromises the availability of the signing operation. If a corrupted node outputs a syntactically valid but semantically incorrect output share, then it may or may not compromise integrity (or the integrity of the final signature), depending on whether or not the mechanism (implicit in the example) responsible for recombining the output shares is prescribed or not to verify the correctness of the signature.

In summary, for the example scheme considered even for the considered simple example of “3-out-of-3” signature scheme (based on a “3-out-of-3” secret sharing), there are different compromise thresholds for different properties: \( f_C = 2 \) for confidentiality; \( f_A = 0 \) for availability; similar to the underlying secret-sharing scheme. For integrity of produced signatures, the compromise threshold is by default also equal to \( f_I = 0 \) or \( f_I = \infty \) (depending on the protocol) for integrity, since the described mechanism produces an incorrect signature if one of the output shares is incorrect. However, one can also consider an analysis that incorporates the context of a signature application where the corruption is detected by a verification against the provided plaintext. If the detection of a bad signature prevents an error propagation in the application, then the integrity compromise can be disregarded \((f_I = n)\) and the problem be instead classified as an availability issue \((f_A = 0)\). More generally, for a “\( k \)-out-of-\( n \)” signature scheme: \( f_C = k - 1 \) for the confidentiality of the signing key; \( f_A \) can depend on the scheme, but ideally can be as high as \( n - k \); \( f_I \) can depend on the scheme and on the application definition of integrity compromise.

It is thus conceivable that based on the above, under certain types of attack the threshold scheme may, in comparison with the conventional scheme, improve the confidentiality of the original key, while degrading the availability and/or integrity of the intended output. Particularly, this happens if: compromising the integrity or (when \( f_A = 0 \)) compromising the availability of one \((= 1 + f_A)\) out of the three nodes in the threshold version is easier than compromising the availability of a conventional non-threshold version; (when \( f_I = 0 \)) if compromising the integrity of one \((= 1 + f_I)\) out of the three nodes in the threshold version is easier than compromising the integrity of a conventional non-threshold version; if compromising the confidentiality in the conventional implementation is easier than compromising the confidentiality of all \( n \) \((= 1 + f_C)\) nodes in the threshold version (when \( f_C = n - 1 \)). In some attack/compromise models it may be possible to quantify the...
likelihood of \( f + 1 \) nodes being compromised, e.g., dependent on an attack intensity and
rejuvenation pattern [BB12]. In particular, one may find that under certain models the
threshold property induces less reliability or availability, e.g., if not properly provisioned
with rejuvenation techniques. If, for example, nodes are of similar type, such as several hardware security modules (HSMs) or several virtual machines (VMs) in
different computers and have diversity at certain levels (OS, vendor, etc.), and if a constant
rate probability of compromise is plausible for certain attack vectors, then it is possible to
analyze the impact of reactive and proactive rejuvenation.

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

4.1.2 A word of caution: pitfalls of decoupling security properties

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

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

Specific attacks. As the above mentioned considerations are also pertinent when changing from a conventional scheme to a threshold scheme. The threshold augmentation may require adjusting an ideal functionality and/or adding definitions and security properties. For example, the communication between components of the threshold scheme may be subject to attacks/compromises that affect security in a way that is not possible in the non-threshold version (where the notion of communication between components is not even applicable). As another example, a security property defined with the help of a “game” (a game-based definition), where an adversary has some access to an “oracle” (e.g., an encryption oracle), may have to update the game definition (including the definition of a success) to account for the possibility of several components being controlled by the adversary.

4.1.3 Specific attacks

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

We have been focusing on attacks against the confidentiality of a key, but attacks can have other goals. An attack focused on breaking availability can try to accomplish a denial of service of some cryptographic operation. The consequences can be catastrophic if the operation is part of a time-sensitive critical mission. Threshold schemes also provide tradeoffs for availability. If nodes can only fail by crash, better availability can typically be obtained by increasing the overall number of nodes \( n \), while keeping \( k \) constant. However, if nodes can become malicious, then availability (of correct operations, i.e., with integrity) requires handling faulty nodes. The cost (in \( n \)) for handling faulty nodes may vary significantly based on the ability vs. inability to detect and replace faulty nodes and may impose restrictions on \( f \) and \( k \).

### 4.1.4 Proofs of Security

Proofs of security are essential in state-of-the-art cryptography. Their importance in supporting proposals of cryptographic schemes is recognized in the “NIST Cryptographic Standards and Guidelines Development Process” document, published by the Cryptographic Technology Group in 2016 [Gro16]. These proofs can serve as a guide to follow a logical path to assess the security of a threshold scheme, being useful to identify assumptions and attack models on which to base security assertions. This can, for example, be helpful to compare security of a threshold scheme vs. the security of a corresponding conventional (non-threshold) scheme. Proofs are characterized by different attributes, e.g.: contextualized to adversary types (e.g., see Sec. 4.2), security parameters (e.g., computational and/or statistical) and other characteristics of a system model (e.g., see Sec. 4.3); proving an enumeration of security properties vs. proving that a scheme (protocol) emulates an ideal functionality; proving security in a standalone setting vs. establishing security under composition with other protocol executions; being thorough in modeling the real world vs. omitting consideration of possible real side-channels. Proofs may also vary with the primitive type and research area, e.g., single-device setting (e.g., threshold circuits) vs. general SMPC.

Overall, results from state-of-the-art research can provide useful insights on these choices.

### 4.2 Types of attack

Security goals are considered with respect to an adversary, also known as an “attacker”. When evaluating a proposal for threshold scheme implementation, we would like to have...
Table 1. Representative attack types

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

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

Static vs. adaptive. In static attacks, the attack pattern, e.g., the choice of which components to try to compromise, does not depend on observations of the protocol execution. In adaptive attacks, the attacker can adapt the adversarial actions based on an observation of the protocol flow. For example, a node may be targeted for intrusion upon being elected to a role of leader in a phase of the protocol.

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

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

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

**Conventional vs. threshold-related.** While threshold schemes may be designed to mitigate the effectiveness of some attacks on conventional applications, the actual implementation of a threshold design may be the cause of new inherent vulnerabilities. For example, an attack may be able to exploit some vulnerability in the communication network that intermediates several nodes, where such a network would not even exist in a conventional implementation. We characterize an attack as threshold-related if the attack vector is inherently allowed by the threshold design. Complementary, there are conventional attacks that can be considered similarly with respect to each component of a threshold scheme. In the latter case, it is still relevant to consider, for example, if an attacker is able to choose whether to attack the nodes/platform in parallel or sequentially.

Tolerance to compromise can be useful even in scenarios of non-intentional adversaries. For example, some systems may be constrained to satisfy auditability requirements that warrant taking down components for audit. If a service is supported on a multi-party threshold scheme with tolerance to compromise, then the audit of components can be done without affecting the overall availability.
The goal of this subsection is to convey possible nuances of system models, in order to encourage a reflection of different consequences they may induce. Several characterizing features of system model for threshold schemes are further discussed in Sec. 5.

**Interactions.**

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

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

A broker can be used to modularize some concerns, e.g., replacing or substantiating usual assumptions, such as the existence of authenticated channels. Depending on the communication model, the broker can, for example, broadcast messages from clients to all components. At the inter-node level, the broker can be a router at the center of a star configuration, substantiating an inter-node (logical) clique model. The broker can also act as a mediator between each client and the set of nodes of the threshold scheme, possibly hiding from the client the threshold scheme layer. For example, the broker can produce secret shares of the client’s messages and then only send these shares to the nodes; in the reverse direction, it can check consistency, and possibly perform error correction, and aggregate replies from a threshold number of nodes, to then just send a consolidated reply to the client. Depending on the protocol, the threshold nature can be hidden or not from the client. Even in the broker
case, the threshold nature of the scheme may, as a feature, be intentionally revealed to the
client. For example, the client may receive a multi-signature enabling non-repudiation of
the participation of a number of nodes in the production of a response.

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

Identity trust.

4.3.2 Identity trust

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

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

A threshold scheme may also be implemented in a setting where the nodes have identities tied to public keys within a public-key infrastructure (PKI). The PKI can then support secure authentication and communication (e.g., with confidentiality and integrity of content and origin) between any pair of nodes. (This assurance assumes that the attacker may control the delivery of messages between nodes but cannot prevent nodes from accessing the root certification authority.) With PKI-based signatures, a threshold scheme can be designed to enable external users to verify that results were indeed obtained upon a threshold interaction.
In a different setting, the initial state of parties might be defined by a joint protocol, e.g., a distributed key generation [Ped92]. The joint computation may yield to every node a share of a new secret, possibly along with authentication credentials. This can conceivably be used by a certification authority (CA) to generate a new signing key, without ever having it available (for leakage) in any localized point. In such case, there is no use for a trusted dealer of shared secrets, although the nodes may still have been deployed by a centralized authority the same entity.

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

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

**Trust between clients and threshold scheme.**

**4.3.3 Trust between clients and threshold scheme**

We have emphasized the use of threshold schemes as a way to enhance the protection of secret keys. But when the threshold system is then used to, say, encrypt or sign messages at the request of a client, is there a concern about confidentiality of the plaintext? An intention to ensure confidentiality of the plaintext may dictate restrictions on the type of threshold scheme and system model. If the plaintext is to remain secret, then the client cannot simply send the plaintext in clear to one or several of the nodes. Alternatively, it may for example: (i) send it through a trusted proxy that creates and sends a corresponding plaintext share to each node; or (ii) it may communicate directly a share to each node; or (iii) it may encrypt shares for each node but send them through a single primary node. Each example may be supported by a nuanced system model, e.g., respectively (i) the existence of a special trusted component; (ii) a communication model where each client can directly communicate with each node; (iii) a PKI (or shared symmetric keys) enabling encrypted communication with each node.
We can also consider the assurances that a client would like to receive from a threshold scheme operation. We already referred to the possibility of a client receiving independent signatures (or multi-signatures) from the nodes. Going further, we can also think of clients wanting to obtain assurance of correct behavior by the nodes. This can be achieved, for example, with the support of publicly verifiable secret sharing (PVSS) schemes [Sta96, Sch99].

Another matter related to the relation between users and threshold system is authentication and authorization of users. Cryptographic modules often have to support an access control mechanism to determine from which users to accept which requests for cryptographic operations. Access control can itself be implemented using a threshold approach.

### 4.3.4 Distributed agreement/consensus

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

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

For example, the problem is solvable even with Byzantine faults if the processes have access to randomness [Ben83, Rab83] or synchronous communication [PSL80, LSP82, DDS87]. In those cases, settings the number of good components must be larger than two-thirds of the total, i.e., $k \geq (2n+1)/3$, or equivalently $n \geq 3f + 1$. Provided the appropriate timing assumptions, if nodes only fail by crash, then a non-crashed simple-majority is sufficient, i.e., $k \geq f + 1$, or equivalently $n \geq 2f + 1$ [Lam06]. In another extreme, consensus is solvable even with a single good party if a suitable trusted setup can be instantiated to enable transferable message authentication. This is the case when a PKI setup enables cryptographic signatures [PSL80], or in some other setups (e.g., reliable broadcast and secret channels in a precomputation phase [PW92]).

The discussion above motivates reflecting also on the property of brittleness [Vas15]. This expresses a degree of susceptibility to a breakdown of the security properties (e.g.,
exfiltration of a key) of a particular algorithm due to errors in the configuration and/or input parameters. In other words, one is concerned with the fragility of a system with respect to changes in the system model or expected setup. Even if a system has all desired properties under a well-defined model, it may be unsuitable for real deployment if it fails catastrophically under reasonable variations of the environment. One would typically prefer instead some kind of graceful degradation. Also related and pertinent is the consideration of how protocols behave differently under different types of attack. Some protocols can be characterized by two (or more) ordered thresholds (e.g., $f_1 < f_2$), meaning that desired security properties hold while the first threshold is not surpassed, but the security failure is not catastrophic while the second threshold is not met [FHHW03]. The thresholds can also depend on the type of attackers, and different nodes can be subject to different types of compromise.

5 Characterizing features

We now provide a high-level structured review of characterizing features of threshold schemes, to facilitate the discussion towards criteria for evaluation of concrete proposals. We intend to motivate a characterization that helps clarify security tradeoffs when reflecting on diverse adversarial models. Put differently, we find that the upfront clarification of certain high-level features is important for discussing the standardization and validation of threshold cryptographic schemes. Table 2 shows examples of possible representations and attributes of characterizing features — see Table 2 — it does not intend to be exhaustive.

5.1 Threshold values

5.1.1 A threshold

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

The considered type of compromise may vary, but we start by focusing simply on threshold numbers. In Correspondingly, the dual threshold notation — $f$ vs. $k$ — enables us to pinpoint each perspective, which can be useful. For example: in some cases, a design goal is directly set as the ability to withstand the compromise of up to a threshold number $f$ of components. In other cases, design constraints such as cost may directly limit the...
total number $n$ of components, which in turn may impose a threshold number $k$ of good components, depending on the protocol and adversarial model.

As already discussed in Sec. 4.1.1, these thresholds (of good and bad number of components) make sense when contextualized (sometimes implicitly) to some security property. For example, when referring to a $k$-out-of-$n$ secret sharing scheme the $k$ refers to availability (minimum number of components necessary to recover a secret), whereas the compromise threshold $f$ for confidentiality of the secret (maximum number of components that together cannot recover the secret) is in that case equal to $k - 1$. The meanings of good and bad and the corresponding thresholds can vary across different security properties. The threshold symbols $k$ and $f$ can be indexed by the corresponding security property (e.g., $f_C$ vs. $f_I$ vs. $f_A$, respectively for confidentiality, integrity and availability), but we omit indices when the context is clear.

### 5.1.2 Relating $n$ vs. $f$ and $k$

<table>
<thead>
<tr>
<th>Feature</th>
<th>Representation</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Threshold type</td>
<td>Threshold numbers of bad ($f$) and good ($k$) nodes</td>
<td>max $f = 0, \ldots, (n - 1)/3$, $(n - 1)/2, n - 1$ or min $k = n, \ldots, 2f + 1, f + 1, 1$</td>
</tr>
<tr>
<td></td>
<td>Variation with security property and attack vector</td>
<td>$(k_{\text{Secrecy}}, k_{\text{Integrity}}) = (1, n)$, $((n - 1)/2, (n - 1)/2), \ldots, (n, 1)$</td>
</tr>
<tr>
<td>Compromise across nodes</td>
<td>common; independent; sequential</td>
<td></td>
</tr>
<tr>
<td>Communication interfaces</td>
<td>Client ↔ crypto module</td>
<td>broadcast; primary node; secret-sharing</td>
</tr>
<tr>
<td></td>
<td>Inter-node structure</td>
<td>star; clique</td>
</tr>
<tr>
<td></td>
<td>Channel protection</td>
<td>TLS; IPSec; dedicated physical connections; trusted paths [NIS01]; application-level encryption</td>
</tr>
<tr>
<td></td>
<td>Target executing platforms</td>
<td>Multiple parties vs. single device</td>
</tr>
<tr>
<td></td>
<td>Software vs. hardware</td>
<td>VMs as components; HSM; crypto accelerators; crypto libraries; trusted computing environments</td>
</tr>
<tr>
<td></td>
<td>Auxiliary components</td>
<td>global clock; proxy; combiner; random number generator (RNG)</td>
</tr>
<tr>
<td>Setup and maintenance</td>
<td>Bootstrap support</td>
<td>dealer; SMPC</td>
</tr>
<tr>
<td></td>
<td>Rejuvenation modes</td>
<td>reactive vs. proactive; parallel vs. sequential</td>
</tr>
<tr>
<td></td>
<td>Diversity generation</td>
<td>offline pre-computation vs. on-the-fly; unbounded vs. limited set</td>
</tr>
<tr>
<td></td>
<td>Diversity levels</td>
<td>operating system; CA; access control; location; vendor; processor architecture; randomization</td>
</tr>
</tbody>
</table>
When analyzing proposals for concrete threshold schemes, we intend that the system model be sufficiently characterized to enable determining allowed relations between \( n \) vs. \( f \) and \( k \). We now compare two examples that illustrate how, Furthermore, it is important to understand how these thresholds can have an extreme variation across security properties.

In Sec. ?? we already showed how a signature scheme, based on a simple example, a \( n \)-out-of-\( n \) secret-sharing scheme, can have secret-sharing scheme has an optimal threshold for confidentiality \((f = n-1, f_C = n-1, i.e., k = 1)\) and at the same time a pessimal threshold for integrity \((f = 0, f_I = 0, i.e., k_I = n)\) and availability \((f_A = 0, i.e., k_A = n)\).

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

The two examples above differ with respect to which properties are optimal vs. pessimal. The integrity threshold was pessimal. In comparison, the two examples have the same availability thresholds \((f_A = 0)\), but different integrity thresholds: pessimal \((f_I = 0)\) in the first example and optimal \((f_I = n-1)\) in the second one. Alternatively example. Furthermore, confidentiality is a relevant property with optimal threshold \((f_A = n-1)\) in the first example, whereas it is not even considered applicable in the second example. The threshold symbols \(k\) and \(f\) could be indexed by the corresponding security property (e.g., \(f_C\) vs. \(f_I\) vs. \(f_A\), respectively for confidentiality, integrity and availability), but we omit indices when the context is clear.

Different thresholds for the same scheme.

5.1.3 Different thresholds for the same scheme

We gave examples for how the same threshold scheme may be characterized by different thresholds for different security properties. Going further, the thresholds may vary even for a fixed qualitative property (e.g., confidentiality, or integrity, or availability). Typically, an active/malicious/byzantine adversary induces a lower fault-tolerance threshold (i.e., lower tolerance to compromise), when compared to a passive and/or crash-only adversary. The
same is true for system model assumptions, such as asynchrony vs. synchrony of communication channels, and the absence vs. existence of a trusted setup such as a public-key infrastructure. The distributed consensus problem in Sec. 4.3.4 shows how a threshold can widely vary depending on the setting.

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

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

Representative questions about a proposed scheme.

5.1.4 Representative questions about threshold values

1. For the desired security properties, what are the threshold values (maximum \( f \) and/or minimum \( k \)), as a function of the total number \( n \) of components? 

2. What envisioned application contexts justify a high threshold for some properties at the cost of a low threshold for other properties (or of other mitigation measures)?

3. How do threshold values vary with respect to conceivable variations of the system model (e.g., synchrony vs. asynchrony, passive vs. active adversaries)?

5.2 Communication interfaces

The augmentation from a conventional cryptographic implementation to a threshold scheme impacts the communication model. Conceivably, a client can now communicate with more than one component (hereafter “node”), and the nodes can communicate between themselves. In Sec. 4.3.1 we already described several nuances of system model, including synchrony vs.

31
asynchrony, and the possible existence of a broker. We now briefly describe three nuances of communication structures related to clients and nodes.

### Client to/from primary node

The client may communicate with the threshold scheme via a single contact component. When such component is one of the $n$ nodes of the threshold scheme, we can call it a primary node for communication. It relays to all other nodes the communication from the client (e.g., a plaintext), and inversely the result (e.g., a signature). For example, it aggregates intermediate results produced by other components, to then send a single consolidated reply to the client. **With a static primary node, the threshold tolerance $\geq f$ would not include the case of communication.** In such a setting the system might, for example with respect to availability, not be able to tolerate the failure of the primary node (if this role does not change across nodes). But other threshold properties, e.g., confidentiality sustained on a secret sharing scheme across all nodes, may remain independent of the use or not of a primary.

### From client to all nodes

If the client is aware of the threshold scheme, it may be able to replicate a request across all components. A possible advantage is ensuring that all correct components receive the same request. Correspondingly, the client may also receive replies from all (or a threshold number of) components and only then decide on a final result. In a different implementation model, the client can perform secret-sharing on the input and then communicate one share per component. This can be used to support confidentiality of the input, e.g., a plaintext to encrypt or sign. At the very least, this prevents components from applying corruptions dependent on the plaintext value. In the reverse direction, the client can reconstruct (possibly with error-correction) an output from a set of replied shares.

### Inter-node communication

---

This publication is available free of charge from: https://doi.org/10.6028/NIST.IR.8214
In typical threshold schemes, the components have to directly communicate between themselves. (An exception is when the client is the sole intermediary between nodes). The inter-node network structure influences the efficiency and security of communication. In a star configuration, a primary node intermediates all communication. In a clique configuration (i.e., a complete graph), all nodes are able to directly contact any other node. For efficiency reasons, a star configuration may be used for most communication and a clique configuration be available for secondary communications. A dynamic selection of the primary node (also known as leader) may enable overcoming cases of it being compromised [CL02].

Representative questions about a proposed scheme.

5.2.4 Representative questions about communication interfaces

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

5.3 Target computing platforms

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

Software vs. hardware.

5.3.1 Software vs. hardware

Cryptography is implemented on a variety of computing platforms. In the early days of the modern technological revolution in computing and communications, cryptographic algorithms were implemented predominantly in hardware. Examples of such embodiments are
the secure phone lines between federal offices in the 1970s. Hardware implementations provide a level of isolation of the sensitive cryptographic keys and their utilization in processing information, along with storage and management of keys and other sensitive parameters. It is natural to think of the physical boundary of a dedicated circuit board, a dedicated chip, a smart card, or USB key. Thus, one can relate that physical boundary to the ideal black box boundary introduced in Sec. 4 and formulate a set of security assertions. This in fact is the foundation for FIPS 140-2 [NIS01], which was initially developed for hardware cryptographic implementations. This standard contains specific security requirements on the physical boundary of hardware modules, namely in Ref. [NIS01, Section 4], which are concerned with ensuring the attacker cannot probe the circuitry and extract the keys. As the adoption of cryptography extended into e-commerce over the Internet, software implementations of cryptography emerged and over the years became a widely used embodiment for cryptographic primitives. Software cryptographic implementations on a general purpose computer (GPC) are just like any other software component that runs within the control of an operating system (OS). GPCs are much more porous (see Sec. 1) and tend to provide fewer assurances with respect to the isolation of cryptographic keys and other security-sensitive parameters from unauthorized access by other applications running on the same GPC/OS platform, or remotely through the network interfaces of the platform. Correspondingly, these software modules are subject only to a subset of the security requirements described in Ref. [NIS01] and are limited to a lower level of security assurances they can claim to deliver. Given this historical context, the distinction of hardware vs. software in FIPS 140-2 comes from the difference in isolation that the approaches provide, and is not directly related to the manner in which the computation is performed. Note, for example, that a Hardware Security Module (HSM) an HSM might actually contain an embedded microcontroller that performs the cryptographic computation in software. Also, some hardware platforms such as a Field-Programmable Gate Arrays (FPGAs) can be “reprogrammed,” a property that was historically reserved for software implementations. For the sake of readability, we will assume a more “traditional” separation between hardware and software, focusing primarily on the isolation properties, rather than on different types of computing platforms.

The hybrid approach to cryptographic implementations aims to benefit from the flexibility in software and the isolation and/or acceleration in hardware. Here a portion of the implementation is in software executing on a GPC/OS platform and another portion is executing on a dedicated HSM attached to the same GPC. Examples of such modules are the Trusted Platform Module (TPM) [Mor11], or the cryptographic extensions of standard CPU instruction sets—Central Processing Unit (CPU) instruction sets, such as the SGX—Software Guard Extensions (SGX) instruction on Intel platforms [Int18], and the TrustZone technology on ARM—Advanced RISC Machine (ARM) processors [ARM18]. These modules can also be used as secure sub-components within a hybrid fault model. The “secure” components have a more restricted mode of compromise (e.g., only by crash), thereby enabling
better thresholds for byzantine fault tolerance of a distributed system composed also of
larger and less secure components [VCB+13, BDK17].

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

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

Another reason to delegate the execution of cryptographic primitives to dedicated
hardware is for performance improvement. An example of this is the AES extension on
Intel [Gue09] and AMD-Advanced Micro Devices (AMD) CPUs [AMD12]. HSMs offer
similar acceleration benefits.

Single-device vs. multi-party.

5.3.2 Single device vs. multi-party

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

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

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

Auxiliary components.

5.3.3 Auxiliary components

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

Representative questions
5.3.4 Representative questions about computing platforms

1. If a proposed threshold scheme is devised for a “single-device” setting, what can go wrong if its components are instead separated and communicate over the Internet?

2. Which parts of the logical boundary of the threshold system do not correspond to a physical boundary, as verified by the system developer or deployer?

3. Is the system simply developed at the software layer, or are there software components tied to particular hardware components?

4. Which auxiliary components support the threshold scheme but have a failure model different from the one applied to the threshold nodes?

5.4 Setup and maintenance

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

Dealer vs. dealer-free setup.

5.4.1 Dealer vs. dealer-free setup

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

In a setting with a dealer, it is relevant to consider the extent to which the protocol security withstands or breaks in the presence of misbehavior by the dealer [BR07]. Some protocols can be made secure against an untrusted dealer, with respect to integrity, if the protocol enables parties to verify correctness of the distributed parameters. Other
protocols may have security hinge on an assumption of a trusted dealer. Depending on the
functionality, there may exist tradeoffs between efficiency and the property of supporting
the malicious dealer.

Rejuvenation of nodes.

5.4.2 Rejuvenation of nodes

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

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

If a compromise is detected, then the corresponding node should be reactively replaced
by a healthy version, lest the system eventually converges to all nodes being compromised.
If the compromises are not detectable but are nonetheless conceivable, then a proactive
recovery should take place. In the threshold signature scheme from Sec. 3, the resharing of
the secret key constitutes a parallel rejuvenation of nodes. If there is no persistent intrusion,
and the number of compromises never exceeds the allowed threshold, then the resharing
brings the whole system back to a pristine state, with all nodes healthy.

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

In some protocols a rejuvenation may have to take place in parallel, e.g., such as the
already discussed example of updating key shares, with all online parties being rejuvenated
simultaneously. In other cases, rejuvenations may occur sequentially, replacing/recovering each node at a time, especially if the process involves a long downtime.

Many of the considerations pertinent to the initial setup of a threshold system are also applicable to the rejuvenation context. For example, is there a “dealer” responsible for setting up the full state of a rejuvenated node or should the state be updated by the set of online nodes?

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

Levels of diversity.

5.4.3 Levels of diversity

Intuitively, a main motivation for threshold schemes is to improve security by withstanding the compromise of some nodes. Yet, a standalone characterization of threshold values does not say anything about the difficulty of compromising the threshold number of nodes. Consider the case of a common vulnerability, i.e., common across all nodes (e.g., a bug in a common operating system). Once the vulnerability is discovered, an adversary might be able to exploit it with negligible cost to compromise all nodes. In this example, this would then be “as easy” as compromising a conventional scheme with the same vulnerability.

Consider an example where all nodes are symmetric with respect to the threshold protocol, i.e., all implement the same functionality. One can then imagine all nodes being implemented in the same manner, say, the same software, possibly containing a common vulnerability. Conversely, each node can also be programmed for the same functionality via different software versions [CA78]. In practice, common vulnerabilities may occur at multiple levels where the set of nodes is homogeneous, e.g., operating system, network protocol, hardware design, physical location, password. Diversity may be implemented across space (i.e., across the components within a threshold protocol) and time (i.e., replacements and executions across time). In the multi-party case, rejuvenation can happen by actually replacing a physical node by a new one. In certain single-device settings, rejuvenation might be limited to refreshing randomness, while the actual hardware structure remains fixed. In a

4 We also bear in mind the possible mapping of threshold properties into side-channel resistance properties.
software setting, rejuvenation may correspond to replacing a virtual machine, or changing
some randomness used when compiling a software version. At some levels, there may be
a small set of variants (e.g., operating systems), whereas others (e.g., passwords) are
impossible to replace.

The use of diversity is a longstanding practice of traditional fault-tolerance, but its use for
security is more intricate [LS04]. Implementation-wise, multiple levels of diversity (among
other properties) may be required to reduce the possibility of common vulnerabilities [SZ05]
and to substantiate an assumption that compromising more nodes is more difficult than
compromising fewer nodes. A fundamental difficulty is that the level of effort used by an
attack vector may be unpredictable until the attack takes place.

Representative questions.

5.4.4 Representative questions about setup and maintenance

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

6 Validation of implementations

6.1 The existing CMVP and FIPS 140-2

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

As we have pointed out, the correct and bug-free implementation of a cryptographic
algorithm and the environment in which it executes are also very important for security. To
assess security aspects related to real hardware and software implementations, NIST estab-
lished the Cryptographic Module Validation Program (CMVP) [NIS18c] in 1995 to validate
cryptographic modules against the security requirements in Federal Information Processing
Standard (FIPS) Publication 140-2 [NIS01]. The CMVP leverages independent third-party
testing laboratories to test commercial-off-the-shelf cryptographic modules supplied by
industry vendors.

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

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

### 6.2 Integration of threshold cryptographic schemes

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

FIPS 140-2 already has security requirements for secret sharing applied to cryptographic
keys. Section 4.7.4 of the standard defines security requirements for split-knowledge
procedures for security levels 3 and 4, stipulating that “if knowledge of $n$ key components
is required to reconstruct the original key, then knowledge of $n - 1$ components provides no
information about the original key, other than the length.” This can for example be satisfied
by implementations of the Shamir and Blakley secret sharing schemes mentioned in Sec. 2.2.

The above-mentioned provision in FIPS 140-2 refers only to secret-sharing and by itself
does not ensure that keys are never recombined when needed by an algorithm, which is
a main subject of threshold schemes for cryptographic primitives. That provision is thus
insufficient to accommodate the plethora of threshold considerations that have been pointed
out in this report. Generally speaking, the process towards standardization of threshold
schemes may involve reconsidering the adequacy of the validation requirements and where
necessary devise new or complementary requirements.

As technology progresses and cryptography becomes ubiquitous in the federal informa-
tion infrastructure, the number and complexity of modules to be validated increases. This makes it increasingly difficult to detect at validation stage all possible defects that might compromise security. This is one more reason to consider the potential of threshold cryptography in avoiding single points of failure in real implementations. However, similarly to conventional cryptography, the security of the threshold cryptographic implementation may also be impacted by defects introduced as a result of human errors or unsafe optimization by the tools used to compile or synthesize the implementation. Thus, it is important to ensure that the algorithms supporting threshold cryptography are theoretically secure, and to verify that they have been implemented correctly. The definition of guidelines would help develop a structured process of formulating and validating security assertions about threshold cryptographic implementations.

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

It is encouraging to note that automated methods for validating protocol implementations have emerged recently (e.g., [CHH+17, BBK17, DLFK+17]). This experience may be useful to leverage for the protocols involved in threshold cryptographic schemes.

7 Criteria for standardization

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

Some representative questions.

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

1. Characterizing features
   (a) Are the *characterizing features* of the threshold scheme fully described?
   (b) On what *executing platforms* can the scheme be implemented?
   (c) What are the *operational costs and properties of setup and maintenance*? What are the *node-rejuvenation* mechanisms (e.g., resharin or node-replacement)?
   (d) What are the *operational costs and properties of setup and maintenance*?
   (e) How are nearby components assumed separate/independent vs. interfering?

2. Applicability of scheme
   (a) How *efficient/performant* are the operations as a function of threshold parameters?
   (b) Is the scheme applicable to *NIST-approved* cryptographic primitives?
   (c) Do *base primitives* (e.g., oblivious transfer) require independent standardization?
   (d) Is the *system model* applicable to known and relevant application contexts?

3. Implementations
   (a) Should the standard take into account *feasibility and interoperability* on different platforms, e.g., hardware or operating systems?
   (b) Should the standard define *common APIs* for client-side functions?
   (c) What degree of automated protocol validation should be targeted for proposed standards?

4. Implementation vs. security
   (a) How is *diversity* of nodes related to known attack vectors?
   (b) Is the implementation complexity likely to lead to *new implementation bugs or misconfiguration*?
   (c) What *trusted setup and assumptions* are required (e.g., dealer, special components)?
   (d) How *brittle* is the scheme (likely to break under small environmental variations)?
   (e) What faults can be detected and reversed, while identifying the culprit node(s)?

5. Security
   (a) What threshold properties relate to resistance against *side-channel attacks* and how?
   (b) Are there identified *security tradeoffs* across attack types and configurations?
   (c) Is the security assessment supported by a *security proof*? How does the *reliability* compare against that of a conventional implementation?
   (d) How *brittle* is the scheme (likely to break under small variations in the environment)?
   (e) What features of *graceful degradation* exist against conceivable failures?
6. Validation

(a) Do the security assertions match / fit into the FIPS 140-2 framework?

(b) How testable is the scheme (can security assertions be tested and validated)?

(c) Is there a proposed automated validation mechanism?

7. Licensing

(a) What are the intellectual property implications and the licensing conditions?

8. New standards development

Depending on the adopted approach to developing the new standards, there may be submissions of candidate threshold schemes for evaluation. If such an approach is adopted then potential criteria for quality submissions might include the following:

(a) Are working implementations available?

(b) Is interoperability between two or more different implementations demonstrated?

(c) Are high-level use-cases/applications (e.g., signing, decryption, etc.) feasible?

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

In addition, there may exist pertinent standardization meta questions, questions about what and how to standardize. What flexibility of parametrization should a standard allow?

Should there be distinct standardization profiles to separately focus on distinct attribute instantiations, e.g., single-device vs. multi-party platform, side-channel attack vs. intrusion per node? Next, we elaborate a bit further on two additional aspects.

7.2 Standardization at what granularity level?

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

While it may be useful to standardize modular components, such development is not on its own sufficient to achieve good standards for threshold schemes for cryptographic primitives. There are difficulties associated with secure composition of secure components into a protocol (e.g., into a threshold scheme). Composability is indeed a recurring subject in the development of secure multi-party computation protocols. It is thus an open question how one should or should not create standards for combining modular components.

Another consideration on what and how to standardize pertains to the potentially large set of available solutions. On the one hand, SMPC provides general techniques that can use common building blocks to enable thresholdizing any cryptographic primitive. On the other hand, there are also specialized (ad-hoc) solutions with techniques tailored to specific primitives and their applications. How should we handle the two types of solutions in the standardization process, both of which are likely to improve over time? Some benchmarking may be helpful for navigating the pool of possibilities and making objective comparisons between ad-hoc and generic solutions. It is important that this process for characterizing such solutions is conducted in a way that invites and encourages participation from all stakeholders.

7.3 Standardization opportunities

At a basic level, secret-sharing schemes can be used to split a secret key while its use is not required, ensuring that a threshold number $f + 1$ of shares is needed to reconstruct the key. However, this by itself does not enable cryptographic primitives to use the key shares instead of the recombined key. Secret sharing alone might also not enable threshold properties for other purposes, such as preventing a corruption of the intended output. The above limitations can be addressed with threshold cryptography (in the broad sense of encompassing threshold schemes for a secure implementation of cryptographic primitives).

For example, the computation may be performed on shares of the key, without the need to ever recombine the original key, and enabling threshold security for other properties, including integrity and availability. What then should be standardized, within the realm of threshold cryptography?

Threshold schemes have wide applicability, in the sense that there are general techniques to convert a conventional implementation into a threshold version thereof. One can thus ask: for which conventional cryptographic schemes should one consider standardization of a threshold version? On one hand, there is a clear interest in enabling threshold versions
of NIST-approved cryptographic primitives. On the other hand, the consideration of standardization of threshold schemes is in itself an opportunity to review suitability for new standards. In this line, we also wonder how the standardization of threshold schemes might also benefit other ongoing NIST efforts of standardization. For example, could elements from the lightweight [MBTM17] and post-quantum cryptography (PQC) [NIS17] projects at NIST be useful for threshold cryptography? Could the schemes considered by those projects also be looked at in the perspective of possible threshold variants? We-More research is needed. That is why we do not intend here to show any preference about concrete cases, but simply to raise the point for consideration. We believe that a better clarification may arise from a constructive interaction with the research community and other stakeholders.

8 Conclusions

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

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

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

Threshold schemes can be efficient. For example, we described how a simple threshold RSA signature scheme based on a $n$-out-of-$n$ secret-sharing has a complexity that increases only linearly with the number of shares, and whose computation is parallelized.
across several nodes. In such case, the simplicity of the method is based on a mathematical
structure-property (a homomorphism) present in the underlying structure of the original
scheme. In contrast, schemes for other cryptographic primitives, such as some block ciphers,
may require significant computational overhead compared to their conventional counterparts.
Still, even in those cases the threshold schemes may be practical and justifiable, depending
on the intended security assurances and the application context.

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

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

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

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

Once criteria are in place, the selection and standardization of concrete schemes should
include an integration with validation methodologies. How then to express security assert-
tions that may fit within FIPS 140-2 or fit well as a complement thereof? What security
and implementation profiles should be devised? When tackling these challenges,
positive synergies may result from engaging with and incorporating feedback from the
research community and other stakeholders.
NISTIR 8214 (DRAFT)

This is a **diff** between the Draft (July 2018) and Final (March 2019) versions. Review comments are included in the end.

### Threshold Schemes for Cryptographic Primitives


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

This is a diff, between the Draft (July 2018) and Final (March 2019) versions. Review comments are included in the end.

NISTIR 8214 *(Draft)*  

**Threshold Schemes for Cryptographic Primitives**

DOI:10.1145/22145.22178.


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

NISTIR 8214 (Draft)  THRESHOLD SCHEMES FOR CRYPTOGRAPHIC PRIMITIVES

2087  2016.
2089
2091
2093
2095
2097
2099
2101
2103
2105
2107
2109
This is a diff between the Draft (July 2018) and Final (March 2019) versions. Review comments are included in the end.

NISTIR 8214 (Draft)  Threshold Schemes for Cryptographic Primitives


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

NISTIR 8214 (Draft)

Threshold Schemes for Cryptographic Primitives


This is a diff, between the Draft (July 2018) and Final (March 2019) versions. Review comments are included in the end.

NISTIR 8214 (Draft)  

Threshold Schemes for Cryptographic Primitives


58
List of feedback comments

The following pages show a table that includes public comments received about the draft published on July 26, 2018. Each source is indexed with an upper-case letter (A, B, ..., M), ordered chronologically by received date. There are 13 distinct sets of comments, plus one set of editorial revisions. Each set of comments was, for the purpose of this revision, separated in individualized items, e.g., A1, A2, ..., B1, ...

Index

- A — Svetla Nikova (KU Leuven) .................................................................................................................. 60
- B — Gokhan Kocak (Asena Inc.) .................................................................................................................. 62
- C — Oliver Stengele (Karlsruhe Institute of Technology) .................................................................................. 63
- D — Aivo Kalu (Cybernetica AS) .................................................................................................................. 64
- E — Christian Cachin, Hugo Krawczyk, Tal Rabin, Jason Resch, Chrysa Stathatkopoulou (IBM, IBM Research) .................................................................................................................. 66
- F — Tanja Lange (Technische Universiteit Eindhoven) .................................................................................. 82
- G — Yehuda Lindell (Bar-Ilan University and Unbound Tech) ........................................................................ 84
- H — Samuel Ranellucci (Unbound Tech) ......................................................................................................... 87
- I — Dan Bogdanov (Cybernetica AS) ............................................................................................................ 89
- J — Rosario Gennaro (The City College of New York) .................................................................................... 97
- K — Thalia May Laing (HP Labs) .................................................................................................................. 101
- L — Karim Eldefrawy (SRI International) .................................................................................................... 104
- M — John Wallrabenstein (Analog) ............................................................................................................. 106
- N — Authors (NIST) – editorial revision ........................................................................................................ 108
<table>
<thead>
<tr>
<th>#</th>
<th>Ref</th>
<th>Old location</th>
<th>A: Comments by Svetla Nikova (KULeuven)</th>
<th>Related</th>
<th>Reply Notes</th>
<th>Rev</th>
</tr>
</thead>
</table>
| 1 | A1  | line 110, last words | “even if one” - why do you write “one” for Example (ii) while Example (iii) has “some”? | A2, E45, E60, F2, K11, K12 | - **NOTE:** They are different examples, to convey a diversity of cases.  
- **CHANGED:** In the last example, changed “some” to “some (but not all)”.
 | R5  |
| 2 | A2  | line 247, lines 983–984 | "f-out-of-n" threshold scheme I believe that this report is the first to introduce the parameter f for the number of nodes that can be tolerated to fail. The literature uses k and n, where the equality k + f = n holds for complete access structures, while incomplete (e.g. ramp not threshold) also exist where k+f <n holds. This notation is explained only on line 983. Line 984 states that the introduction of f is useful, but the document does not make clear how. At least, the notation should be defined before it is uses. I recommend to be careful with the terminology "f-out-of-n threshold scheme", because readers might confuse it with the more commonly used term "k-out-of-n threshold scheme". | A2, E145, E60, F2, K11, K12 | - **NOTE:** The compromise-threshold parameter (f or some other letter) has been used in the literature, e.g., see Refs. [Lam06] and [GRJK00]. The subsequent paragraph (old lines 986–989) conveyed how both perspectives can be “useful”.  
- **CHANGED:** We have now added several clarifications about the co-existing dual (f vs. k) notation, e.g.: in section 1, new paragraph to mention k; old Sec 2.1 (“secret sharing”, now Sec. 2.2) now differentiates f and emphasizes k; old subsection 2.5 (“terminology”, now Sec. 2.1) now explicitly mentions the dual perspective; in Sec. 5.1.1, a slight text adjustment to bring “useful” closer to its explanation.
 | R3, R13, R17, R18, R49, R74, R72 |
| 3 | A3  | Lines 714, 716, 719 | you use f_A, f_I, f_C but define them only on line 1012 | A3, E61 | - **NOTE:** The symbols had been introduced in old lines 709–710 in Sec. 4.1, explaining that they correspond to different compromise thresholds for different properties.  
- **CHANGED:** Edited the mentioned paragraphs to clarify the index notation for different f thresholds. Move up (to Sec. 5.1.1) the note that clarifies the possible omission of indices.
 | R51, R52, R53, R73 |
| 4 | A4  | Line 399-407 | is it useful to introduce the (P,omega) notation in this report? Is it useful to mention an issue that is not a concern? | - **NOTE:** The angular (\(\omega\)) notation is used to convey complementary intuition for who may appreciate the geometric aspect of line rotation over a point (P).  
- **CHANGED:** Removed “concern”; simplified description.
<p>| R29  |</p>
<table>
<thead>
<tr>
<th>#</th>
<th>Ref</th>
<th>Old location</th>
<th>A: Comments by Svetla Nikova (KULeuven)</th>
<th>Related</th>
<th>Reply Notes</th>
<th>Rev</th>
</tr>
</thead>
</table>
| 5 | A5  | Line 622     | [Ber05]: this paper discusses the effects of key-dependent cache hits vs. cache misses. I think this can also be called a memory access, so the statement that it is not sufficient to have source code without key dependent memory accesses is a bit dubious. Note also that the "remote attack" described in [Ber05] requires a malicious program to run locally and communicate with a remote host. The attack as described requires perfect knowledge of all software and libraries running on the system, in the correct versions. I think that the following is a better reference: Eran Tromer, Dag Arne Osvik, Adi Shamir: Efficient Cache Attacks on AES, and Countermeasures. J. Cryptology 23(1): 37-71 (2010) | A5, E57 | – **NOTE:** The old subsequent paragraph already explains/cites two use-cases where constant-time code may be insufficient.  
– **CHANGED:** For clarification: adjusted the position of the mentioned citation in the sentence; merged this to the subsequent paragraph, which already contained several references that justify the statement. | R45 |
| 6 | A6  | Line 1207    | suggestion for an additional representative question: If a threshold scheme is proposed for a single-device setting, what assumptions does it make about the separation/independence of the "parties" that in practice sit next to one another in the device? | A6, E19 | – **NOTE:** The separation between parties is also a relevant matter in the single-device setting.  
– **CHANGED:** Added a related question. | R93 |
| 7 | A7  |              | The report mentions repeatedly that k-out-of-n threshold schemes, in particular when k is high, might lead to a lower security level than a 1-out-of-1 scheme. This is correct and needs to be mentioned, but I think it deserves less emphasis. In my opinion, the issue mentioned above is much more critical to assess the practical security of a threshold scheme. | A7, G4, E14, E16, J11 | – **NOTE:** We find essential to highlight that there is a multitude of security properties that can be differently affected by a threshold augmentation. For example, if k = n then availability may be degraded. We agree that there are other critical issues.  
– **CHANGED:** No direct change, but see the reply to E14. | — |
<table>
<thead>
<tr>
<th>#</th>
<th>Ref</th>
<th>Old location</th>
<th>B: Comments by Gokhan Kocak (Asena Inc.)</th>
<th>Related</th>
<th>Reply Notes</th>
<th>Rev</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>B1</td>
<td></td>
<td>I would like to share my observations about secret sharing schemes:</td>
<td>B1, G2, I2, M1</td>
<td>– <strong>NOTE:</strong> This NISTIR intends to promote discussion, including with industry, about threshold schemes for cryptographic primitives, including those based on secret sharing schemes. – <strong>CHANGED:</strong> No change.</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- It’s not publicly known</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Even the IT Industry has limited knowledge about the secret sharing schemes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Storing pieces of data at different places is not an IT tradition, IT should accept this change and adapt to it.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>B2</td>
<td></td>
<td>We have implemented a threshold secret sharing using the following method:</td>
<td></td>
<td>– <strong>NOTE:</strong> The process of standardization should lead to clear descriptions of threshold schemes. – <strong>CHANGED:</strong> No change.</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Create a random key of sufficient length</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Create a random nonce to be used in AES encryption</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Encrypt the input using the random key</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Create 3 pieces of the encrypted input using 2 out of 3 threshold scheme</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Store the AES key, nonce, hash of the original input, hash of the encrypted output in a data structure of 512 bytes. Also add some padding with random data.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- The hashes are used to make sure that the original input is created after combining pieces</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Create 3 pieces of this data structure using 2 out of 3 threshold scheme</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Try all possible combinations to re-create the original input in memory (i.e. one key is lost, one piece of data is lost, key 1 and key 3 are available, key 1 and key 2 are available etc)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Store all the pieces at different directories</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Have the user distribute these pieces to different locations</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Algorithms we used:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Shamir’s Secret Sharing algorithm</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- AES256 encryption</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- SHA256</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>B3</td>
<td></td>
<td>We encrypted the original input because if someone finds 2 pieces of it, then he/she can re-create the encrypted data but not the original data. Because he/she needs the encryption key and nonce to decrypt it. We stored the hashes of input data and encrypted data in our key file together with AES key and nonce. After re-creating the original data, we used these hashes to check if we really created the original data. It’s very important to keep the pieces of data and pieces of keys at different locations.</td>
<td></td>
<td>– <strong>NOTE:</strong> New standards of threshold schemes for cryptographic primitives should include techniques that enhance security. – <strong>CHANGED:</strong> No change.</td>
<td>—</td>
</tr>
</tbody>
</table>
I am a scientific staff member within the research group "De-centralized Systems and Network Services" at the Karlsruhe Institute of Technology (KIT) and I would like to offer a comment on your Draft NISTIR 8214 titled "Threshold Schemes for Cryptographic Primitives".

My current research focuses on using Threshold Encryption schemes to manage the flow of information in processes between mutually distrusting parties [2] using a decentralized consensus system like Ethereum [1]. Other researchers are using Threshold Encryption and Threshold Signature schemes, both with and without a trusted dealer, for other applications in the context of distributed systems [3,4].

By their base functionality, Threshold Schemes appear destined to become a vital component of various distributed and decentralised processes. I would recommend that you keep this burgeoning field of possible applications in mind during the standardisation process.

<table>
<thead>
<tr>
<th>#</th>
<th>Ref</th>
<th>Old location</th>
<th>D: Comments by Aivo Kalu (Cybernetica)</th>
<th>Related</th>
<th>Reply Notes</th>
<th>Rev</th>
</tr>
</thead>
</table>
| 12 | D1  |              | First, I would like to thank NIST for preparing the report on the threshold cryptography subject. This is a promising field with many possible applications and we are very interested to exchange views and have a discussion with you. | D1, E1, F1, G1, H2, H8, I1, J1, K1, L1, M2 | – **NOTE:** The endeavor has its complexities.  
– **CHANGED:** No change. | — |
| 13 | D2  |              | I would wish to highlight the possibility of the multi-prime RSA approach, which enables some rather interesting applications. As far as we know, multi-prime RSA is technically compliant with the RFC8017 definition and when the individual primes are generated according to the sizes and algorithms specified in the NIST FIPS 186-4, appendices B3, C.3, and F, it can be as secure as the two-prime RSA as well. |   | – **NOTE:** The steps towards standardization of threshold schemes may analyze various approaches.  
– **CHANGED:** No change | — |
| 14 | D3  | Line 567    | So, if we may propose a specific snippet to be added to the NISTIR text as well, the following place looks like a good candidate: Page 11, lines 567: "Different kind of approach, using multi-prime RSA, is suggested by [https://www.researchgate.net/publication/297585554_On_the_Security_of_Distributed_Mulprime_RSA] and [https://www.researchgate.net/publication/319071255_Server-Supported_RSA_Signatures_for_Mobile_Devices], where the individual shares are generated as independent RSA key pairs and the public moduli of the shares are then multiplied together. This results in the dealer-less RSA key generation, without often expensive zero-knowledge proofs or SMPC protocols. |   | – **NOTE:** In this paragraph we intended a close relation to the RSA example given in the beginning of section 3.1, which defined the modulus \( N \) as a product of only two primes.  
– **CHANGED:** Added a brief note about different RSA-based schemes allowing various tradeoffs, including related to properties of the modulus factorization. | R41 |
| 15 | D4  |              | Also, the row "Bootstrap support" in the table 2 on the page 23 should be amended, to include the "multi-prime RSA" as well. | D4, E21 | – **NOTE:** Table 2 conveys several examples while, for the most part, keeping abstract from concrete primitives. For example, not even the two-prime RSA was included there.  
– **CHANGED:** In the first paragraph of Sec. 5, clarified that Table 2 does not intend to be exhaustive. | R67 |
Another interesting issue with SplitKey signature scheme is the following consideration: At certain abstraction level, we are simply doing 2-out-of-2 or 3-out-of-3 threshold RSA signature scheme. However, the usual academic literature about the threshold schemes tends to assume that nodes, which hold the shares of the private key are independent and deploy the independent access control mechanism for consenting to the operation as well. Therefore, the "compromise" of the node might mean that the access control mechanism is compromised and attacker can therefore use the share to perform the crypto operation, even when he doesn’t have the clear-text copy of the share.

-- NOTE: See reply to item D8, about access control.
-- CHANGED: See reply to item D8.

However, in my view, one of the important aspects of the SplitKey approach is that we have abstracted this 2-out-of-2 threshold access control of two nodes to the single end-user and we are providing him/her with a convenient way to use multiple authentication factors to control all nodes and this way, all shares. So, by entering the PIN (which will be used do decrypt the user’s share) and by using the one-time-password (which is bound to the mobile device by updating it for each next operation) the access control to the local share and the shares, which are stored on the server-side, is achieved. Because this access control mechanism is interleaved tightly with keys and signatures, there’s no way for the server to bypass this mechanism.

-- NOTE: See reply to item D8 about access control.
-- CHANGED: See reply to item D8.

So, it may be difficult to explain the SplitKey signature scheme with all of the proposed characterizing features of Table 2 of NISTIR.

Diversity levels: D7 (AC), E16 (OS), E62 (CA), E71 (ref), E72 (vendors).

-- NOTE: Table 2 conveys examples of possible attributes of characterizing features, but does not intend to be exhaustive.
-- CHANGED: In Table 2, added a new row for “diversity levels”, including “access control” as an example.

In the section 4.3 - System model you have already covered quite a lot of the details of the possible threshold systems. I would propose that the page 19, lines 841 - 851 also cover the possible ways to authenticate the users of the threshold system. It looks like that SplitKey approach, where the authentication is bound with some of the cryptography itself, may result a more secure system.

-- NOTE: Authentication between users and the threshold system is also a pertinent concern.
-- CHANGED: Several paragraphs below, under the “Trust between clients and threshold scheme” (within Sec. 4.3.3), added a short paragraph mentioning authentication, authorization and that an “access control” mechanism can also be thresholdized.
<table>
<thead>
<tr>
<th>#</th>
<th>Ref</th>
<th>Old location</th>
<th>E: Comments by Christian Cachin; Hugo Krawczyk; Tal Rabin; Jason Resch; Chrysoula Stathakopoulou (IBM)</th>
<th>Related</th>
<th>Reply Notes</th>
<th>Rev</th>
</tr>
</thead>
</table>
| 20 | E1  |              | We represent a group of researchers and practitioners within IBM who are working to create an open platform for threshold cryptography. We have many years of joint experience in developing threshold schemes and related technology. We want to extend our gratitude to you for starting this initiative to standardize threshold cryptography. We view threshold cryptography as an important tool in the fight against threats that face information systems. Standardizing the concepts, terminology, and metrics of threshold secure systems is a crucial first step in the path towards greater adoption of these techniques. | D1, E1, F1, G1, H2, H8, I1, J1, K1, L1, M2 | – **NOTE**: Thank you for the encouragement.  
– **CHANGED**: No change. | – |
| 21 | E2  |              | In this shared goal of more widespread adoption of threshold cryptography, we hope to provide you with constructive feedback regarding the draft; a document we found to be impressively thorough and comprehensive. In this draft, you have put together a truly great and useful document. Thank you! |          | – **NOTE**: See E1.  
– **CHANGED**: No change. | – |
| 22 | E3  |              | Our high-level feedback is included below within this e-mail. It is broken down into one of six categories:  
Implementation  
Security Model  
Communications Channels  
Algorithms and Functions  
Post-Quantum Security  
Topics for Further Elaboration  
In addition to these high-level feedback, we have also attached to this e-mail a document listing a larger number minor items of feedback. |          | – **NOTE**: See comment items E4–E40; to which will follow additional items E41–E72  
– **CHANGED**: No change. | – |
<table>
<thead>
<tr>
<th>#</th>
<th>Ref</th>
<th>Old location</th>
<th>E: Comments by Christian Cachin; Hugo Krawczyk; Tal Rabin; Jason Resch; Chrysoula Stathakopoulou (IBM)</th>
<th>Related</th>
<th>Reply Notes</th>
<th>Rev</th>
</tr>
</thead>
<tbody>
<tr>
<td>23</td>
<td>E4</td>
<td>Line 35</td>
<td>Implementation In the introduction (line 35), it is mentioned that the Information Technology Laboratory develops proof of concept implementations. In the area of proof-of-concept implementations, we wanted to make you aware of our open source project (currently hosted at <a href="https://github.com/jasonkresch/pross/">https://github.com/jasonkresch/pross/</a>) which is an effort to build a back-end platform for threshold-secure distributed key generation, proactive refresh, share reconstruction, and rekeying of shareholders. Our aim is to extend its functionality with client-side protocols to enable threshold signing, decryption, password protected secret sharing, pseudo-random functions (including oblivious PRF), changes to K-of-N parameters, and other enhancements. It relies on a Byzantine-Fault Tolerant atomic broadcast channel provided by the BFT-SMaRt library.</td>
<td>Note: Proof of concept implementations are of interest for this project.</td>
<td>Changed: No change.</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>E5</td>
<td></td>
<td>We view implementations as an important and necessary component of any standardization effort. Therefore, we consider it important to define requirements of implementations for which standards will be built. Some important questions to consider may include:</td>
<td>Note: Implementations are important.</td>
<td>Changed: No change. See items E6–E12.</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>E6</td>
<td></td>
<td>Should standards submitters be required to provide working implementations?</td>
<td>Note: Relevant.</td>
<td>Changed: Representative question 8a.</td>
<td>R103</td>
</tr>
<tr>
<td>26</td>
<td>E7</td>
<td></td>
<td>Should standards submitters demonstrate interoperability between two different implementations?</td>
<td>Note: Relevant.</td>
<td>Changed: Representative question 8b.</td>
<td>R104</td>
</tr>
<tr>
<td>27</td>
<td>E8</td>
<td></td>
<td>Must implementations be open source, or should open source be viewed as a strong advantage?</td>
<td>Note: Representative question 7a about licensing and intellectual property is general enough to cover many licensing models, including open-source.</td>
<td>Changed: No change.</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>E9</td>
<td>3a</td>
<td>Should implementations be designed for interoperability between hardware or operating systems? (for improved component diversity)</td>
<td>Note: Relevant.</td>
<td>Changed: Representative question 3a.</td>
<td>R94</td>
</tr>
<tr>
<td>29</td>
<td>E10</td>
<td></td>
<td>Should the standard define common APIs for client-side functions (or should standards focus on back-end server inter-communication) – or might they be modular supporting a mix of different front- and back-end protocols?</td>
<td>Note: Relevant.</td>
<td>Changed: Incorporated in the text.</td>
<td>R95</td>
</tr>
<tr>
<td>#</td>
<td>Ref</td>
<td>Old location</td>
<td>E: Comments by Christian Cachin; Hugo Krawczyk; Tal Rabin; Jason Resch; Chrysoula Stathakopoulou (IBM)</td>
<td>Related</td>
<td>Reply Notes</td>
<td>Rev</td>
</tr>
<tr>
<td>---</td>
<td>----</td>
<td>--------------</td>
<td>-------------------------------------------------</td>
<td>--------</td>
<td>-------------</td>
<td>-----</td>
</tr>
</tbody>
</table>
| 30 | E11 | | What degree of automated protocol analysis should be targeted for proposed standards? (During the TLS 1.3 standardization, many tools were used to prove protocols and implementations, including https://tamarin-prover.github.io, http://prosecco.gforge.inria.fr, and https://www.microsoft.com/en-us/research/project/project-everest-verified-secure-implementations-https-ecosystem/) | E11, I8 | ~ **Note:** Relevant.  
~ **Changed:** Incorporated in the text. | R91, R96 |
| 31 | E12 | | Should implementations demonstrate at least one high-level application (e.g. signing, decryption, PRF, etc.) | | ~ **Note:** Relevant.  
~ **Changed:** Incorporated in the text. | E12 |
| 32 | E13 | Security Model | There is much discussion throughout the document on the topic of the security model of the threshold system and how it is related to the diversity of common components. On this topic we had a few high-level comments. | | ~ **Note:** Security model and diversity are important topics.  
~ **Changed:** See items E14–E18. | — |
| 33 | E14 | Lines 117–119 | In lines 117-119, the topic of whether in some cases using a distributed threshold system might increase an attack surface or otherwise might make some attacks easier than against a conventional (non-threshold) primitive. A chief example might be comparing a threshold system to a single location of a Hardware Security Module (HSM). HSMs are designed to make compromise as difficult as possible, and due to the limited functionality of most HSMs, one might have to choose between using a threshold system and using an HSM. | A7, G4, E14, E16 | ~ **Note:** Depending on the attack/compromise model, the security degradation can even happen if each of the components in the threshold scheme (e.g., several HSMs) is as resilient as the single component in the conventional scheme (e.g., a single HSM). The example suggested in the comment, comparing one HSM vs. a threshold system composed of several weaker components, and where the latter may be more secure, is a different (and also relevant) consideration.  
~ **Changed:** Added text to the end of the paragraph, to enable considering the two cases. See also the reply to E16. | R7 |
| 34 | E15 | | One thing that may be useful to highlight is that a number of off-the-shelf HSMs, which support only standard cryptographic functions, such as RSA signing, or ECDH key derivation, can in many cases be repurposed to perform threshold cryptography natively. This means for certain choices of threshold cryptography, one can get the advantages of threshold cryptography together with the high security of keeping shares only within an HSM. | E15, G4 | ~ **Note:** The 3-out-of-3 threshold RSA-signature example shows a case where each component does the same cryptographic operation as in a conventional operation. An actual threshold scheme may have to further include coordination between parties.  
~ **Changed:** No change. | — |
<table>
<thead>
<tr>
<th>#</th>
<th>Ref</th>
<th>Old location</th>
<th>E: Comments by Christian Cachin; Hugo Krawczyk; Tal Rabin; Jason Resch; Chrysoula Stathakopoulou (IBM)</th>
<th>Related</th>
<th>Reply Notes</th>
<th>Rev</th>
</tr>
</thead>
</table>
| 35 | E16 | Using a system of K-out-of-N HSMs, from different HSM vendors, can also simplify the analysis of a security model. For example, by making some classes of operating system vulnerabilities irrelevant to the exposure of a share that is held on the HSM. One could construct a security model, borrowing from the ideas of reliability analysis where a "mean time to exposure" of a thresholdized secret could be derived from an estimated "mean time to exploit" and "mean time to patch" of the constituent components, assuming each of the N components (here an HSM) has an independent probability of an exploit being discovered and a fixed average time for a patch being issued and applied. In the case where leakage is silent, (undetectable), this greatly amplifies the advantages of proactive refreshing. | Threshold security: A7, G4, E16. Diversity levels: Diversity levels: D7 (AC), E16 (OS), E62 (CA), E71 (ref), E72 (vendors). | ~ **Note:** Models can be used to compare the role of reactive and proactive refreshing.  
~ **Changed:** In Sec. 4.1, added sentence mentioning that an analysis that models nodes of similar type (HSM, or other) having diversity at certain levels (e.g., OS, vendor), can help distinguishing the impact of diverse models of rejuvenation (R54). In Sec. 5, added new row “diversity level” in Table 2, including examples “vendors” and “operating systems”, among others (R71). | R54, R71 |
| 36 | E17 | Secure Enclaves (e.g. SGX) are mentioned in the paragraph on lines 1125 - 1135. Perhaps these "secure enclaves" should be mentioned in Table 2 on page 23 as an example of software vs. hardware. Due to the limited programmability of most HSMs, secure enclaves (or any other trusted computing environment) provide a much needed layer within which to implement advanced protocols and functionality. While HSMs can hold and operate upon shares using PKCS11 functions, there is no built-in functionality for distributed key generation, proactive refresh, or share reconstruction. In non-programmable HSMs, this functionality must exist outside the confines of the HSM. | E15, I19 | ~ **Note:** Intermediate layers and hybrid instantiations enable potential solutions.  
~ **Changed:** Added example “trusted computing environments” to the mentioned row of Table 2. | R69 |
| 37 | E18 | A final comment about the security model, is that there was limited discussion regarding the need for a trusted setup in these solutions. For example, in Pedersen’s distributed key generation, there is a parameter "h" which no one should know the discrete logarithm of. Setting up such "nothing up my sleeve" numbers is a critical point of discussion in some threshold protocols. | E15, I19 | ~ **Note:** Section 3.1 already highlights differences between having a dealer and not having a dealer. Representative question 4c asks about trusted setup.  
~ **Changed:** No direct change, but see new representative question 4c relating trusted setup and proofs of security. | R101 |
<table>
<thead>
<tr>
<th>#</th>
<th>Ref</th>
<th>Old location</th>
<th>E: Comments by Christian Cachin; Hugo Krawczyk; Tal Rabin; Jason Resch; Chrysoula Stathakopoulou (IBM)</th>
<th>Related</th>
<th>Reply Notes</th>
<th>Rev</th>
</tr>
</thead>
<tbody>
<tr>
<td>38</td>
<td>E19</td>
<td>Sec. 1</td>
<td>Communications Channels Among other things, we were very pleased to see the level of attention that this document dedicates to considerations of the communications channel and how that can impact the security properties of the resulting threshold system. We consider it crucial enough to merit some discussion in the introduction (pages 1-3) as the attack models and realistic schemes differ enormously whether communication and the N components are (a) modules on the same chip; (b) components within the same server/machine; (c) servers in the same data center; (d) servers linked only by a wide-area network.</td>
<td>A6, E19, E20, E21</td>
<td>– NOTE: Attack models and security properties can vary substantially with the type of implementation platform and communication between nodes – CHANGED: In Sec. 1 (Introduction), mentioned this example after enumerating characterizing features.</td>
<td>R15</td>
</tr>
<tr>
<td>39</td>
<td>E20</td>
<td>Lines 874–885</td>
<td>This could be discussed more in section 4.3. The paragraph on lines 874-885 contains such a discussion that touches on these considerations, but it seems so important that it should be promoted to its own paragraph or subsection.</td>
<td>E19, E20, E21, A6</td>
<td>– NOTE: Sec. 4.3 motivates the need to describe system models; the later Sec. 4.3 discusses further aspects of its features. – CHANGED: For easier referencing the old paragraphs titles in subsection Sec. 4.3 were promoted to subsubsections with their own numbering.</td>
<td>N5</td>
</tr>
<tr>
<td>40</td>
<td>E21</td>
<td>Table 2</td>
<td>Some of the considerations also appear in Table 2 on page 23, but it is not exhaustive; the multiple parties vs. single device target executing platform lists only three possible deployments.</td>
<td>D4, E21</td>
<td>– NOTE: Table 2 does not intend to be exhaustive. Sec. 5.3.2 mentions a few other examples. – CHANGED: In the first paragraph of Sec. 5, clarified that Table 2 does not intend to be exhaustive.</td>
<td>R67</td>
</tr>
<tr>
<td>41</td>
<td>E22</td>
<td>Sec. 5</td>
<td>Overall, however, we find this section 5 &quot;Characterizing features&quot; to be highly useful and important.</td>
<td></td>
<td>– CHANGED: No change.</td>
<td>—</td>
</tr>
<tr>
<td>42</td>
<td>E23</td>
<td>Lines 880–885</td>
<td>We have very strong agreement with characterizing parameters listed, as starting on line 880. We think it is important to stress that many protocols described in the literature make assumptions (e.g. synchronized clocks or synchronous broadcast channels) that can be impractical or unrealistic in some deployments, such as over the Internet (which is more adequately modeled as an asynchronous network), and failing to consider such subtleties can result in complete breaks of the protocol’s security.</td>
<td></td>
<td>– NOTE: Old lines 969–972 already mentioned that “if a system has all desired properties under a well-defined model, it may be unsuitable for real deployment if it fails catastrophically under reasonable variations of the environment” (Sec. 4.3.4). – CHANGED: No change.</td>
<td>—</td>
</tr>
<tr>
<td>#</td>
<td>Ref</td>
<td>Old location</td>
<td>E: Comments by Christian Cachin; Hugo Krawczyk; Tal Rabin; Jason Resch; Chrysoula Stathakopoulou (IBM)</td>
<td>Related</td>
<td>Reply Notes</td>
<td>Rev</td>
</tr>
<tr>
<td>----</td>
<td>-----</td>
<td>--------------</td>
<td>-----------------------------------------------------------------------------------------------</td>
<td>---------</td>
<td>-------------</td>
<td>-----</td>
</tr>
<tr>
<td>43</td>
<td>E24</td>
<td>Lines 1152...</td>
<td>On page 28, and the paragraph starting on 1152: another section should be inserted here, discussing the communication among the nodes (for example, using communication means (a)–(d) mentioned above).</td>
<td>E19</td>
<td></td>
<td>—</td>
</tr>
<tr>
<td>44</td>
<td>E25</td>
<td>Lines 158, 1423</td>
<td>Algorithms and Functions In several places, both lines 158 and 1423, the document expresses interest in enabling threshold versions of NIST-approved cryptographic primitives. We think it is important to consider also motivations flowing from the other direction. That is, when there are standards for threshold cryptography, NIST ought to consider standardizing cryptographic primitives that are threshold friendly (for example Schnorr and BLS signatures, ECIES decryption).</td>
<td>E25, E27, G2</td>
<td></td>
<td>—</td>
</tr>
<tr>
<td>45</td>
<td>E26</td>
<td></td>
<td>DSA/ECDSA, in particular has a number of properties which make it difficult or inefficient to thresholdize. (See <a href="https://eprint.iacr.org/2016/013.pdf">https://eprint.iacr.org/2016/013.pdf</a> for a treatment of this topic). ECDSA is increasingly replacing RSA, but no fully satisfactory threshold solution exists for it.</td>
<td></td>
<td></td>
<td>—</td>
</tr>
<tr>
<td>#</td>
<td>Ref</td>
<td>Old location</td>
<td>E: Comments by Christian Cachin; Hugo Krawczyk; Tal Rabin; Jason Resch; Chrysoula Stathakopoulou (IBM)</td>
<td>Related</td>
<td>Reply Notes</td>
<td>Rev</td>
</tr>
<tr>
<td>----</td>
<td>-----</td>
<td>--------------</td>
<td>-------------------------------------------------</td>
<td>---------</td>
<td>-------------</td>
<td>-----</td>
</tr>
<tr>
<td>46</td>
<td>E27</td>
<td>Line 1398</td>
<td>In terms of the request on line 1398, &quot;to develop an objective criteria set to support a call for and a selection of schemes for standardization&quot;, we list below a set of schemes that are both simple and efficient to thresholdize: Distributed Secret Generation, Storage, Retrieval Long term storage of keys, or short small secrets Threshold PRF More secure/trusted/harder to bias PRF, KeyStream generation Threshold (Partially) Oblivious PRF Password protected secret sharing (<a href="https://eprint.iacr.org/2017/363">https://eprint.iacr.org/2017/363</a>), password databases (<a href="https://eprint.iacr.org/2015/644.pdf">https://eprint.iacr.org/2015/644.pdf</a>), key management systems (<a href="https://eprint.iacr.org/2018/733">https://eprint.iacr.org/2018/733</a>), password management (<a href="https://eprint.iacr.org/2018/695.pdf">https://eprint.iacr.org/2018/695.pdf</a>), PAKEs (<a href="https://eprint.iacr.org/2018/163.pdf">https://eprint.iacr.org/2018/163.pdf</a>) Threshold (RSA, Schnorr, BLS) Signatures Distributed Certificate Authorities Threshold Bilinear Pairing Identity based encryption, BLS signatures, Threshold Partially Oblivious PRFs Threshold (RSA, El Gamal, ECIES) Decryption Voting schemes, long term storage of sensitive documents to be later published (e.g. wills, audit records).</td>
<td>E25, E27, G2</td>
<td>– <strong>NOTE:</strong> We hope to hear from stakeholders about recent and other threshold schemes. The representative questions in Sec. 7.1 suggest several aspects to considered when discussing those schemes. – <strong>CHANGED:</strong> No change.</td>
<td></td>
</tr>
<tr>
<td>47</td>
<td>E28</td>
<td></td>
<td>Notably, some of these schemes do not even have non-threshold (1-of-1) standardization.</td>
<td>E25</td>
<td>– <strong>NOTE:</strong> See reply to item E25 – <strong>CHANGED:</strong> No change.</td>
<td></td>
</tr>
<tr>
<td>48</td>
<td>E29</td>
<td></td>
<td>There is also the matter of standardizing the maintenance operations (protocols to refresh and reconstruct shares over time).</td>
<td></td>
<td>– <strong>NOTE:</strong> “Maintenance” is identified in a main domain of characterizing features (see “Setup and Maintenance” in Sec. 5.4), namely including considerations about “rejuvenation of nodes” Sec. 5.4.2. – <strong>CHANGED:</strong> No change.</td>
<td></td>
</tr>
<tr>
<td>49</td>
<td>E30</td>
<td></td>
<td>We think it would be helpful to include a section for discussing various applications of different threshold operations; at a minimum it could serve to motivate further research and adoption.</td>
<td>C1, I2, I5, E30, H7 (higher-level apps)</td>
<td>– <strong>NOTE:</strong> Applications can be motivating. We hope to hear more from stakeholders. – <strong>CHANGED:</strong> See reply to I5.</td>
<td>R12</td>
</tr>
<tr>
<td>#</td>
<td>Ref</td>
<td>Old location</td>
<td>E: Comments by Christian Cachin; Hugo Krawczyk; Tal Rabin; Jason Resch; Chrysoula Statathopoulos (IBM)</td>
<td>Related</td>
<td>Reply Notes</td>
<td>Rev</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
</tbody>
</table>
| 50 | E31 | Line 1427, Sec. 7 | Post-Quantum Security On line 1427 there is a brief treatment of post-quantum cryptography. On this topic we believe NIST could be enormously helpful in motivating research into post-quantum threshold-secure, cryptography. There has been some research into threshold systems for lattice-based cryptography (https://eprint.iacr.org/2009/391.pdf, https://www.tau.ac.il/tromer/papers/tfhe-mpc.pdf) as well as a recent "Universal Thresholdizer" https://eprint.iacr.org/2017/251. However, more research is required in this area for constructing threshold systems based on PQ primitives such as lattices, isogeny cryptography, etc. | E31, E32 | – **NOTE:** The consideration of PQ threshold schemes is natural, since both PQ schemes and threshold schemes are developing interests.  
– **CHANGED:** In Sec. 7.3, in the corresponding paragraph, we added a short note: "More research is needed". | R111 |
| 51 | E32 | | One point which would be prudent to consider now is whether PQ algorithms to be selected by NIST for standardization should include "efficient thresholdizability" as a desirable criterion. | E31, E32 | – **NOTE:** See reply to item E31.  
– **CHANGED:** No change. | R111 |
| 52 | E33 | | Topics for Further Elaboration  
This section covers various areas, and is meant to highlight areas we thought could benefit from more elaboration. | E34–E40 | – **NOTE:** See items E34–E40.  
– **CHANGED:** No change. | — |
| 53 | E34 | | While much time is spent discussing RSA, there is no mention of elliptic curves (or Diffie-Hellman-type schemes) which use math and keys that are efficient and straightforward to apply in a threshold scheme. More modern algorithms and techniques, which are also threshold-friendly, such as bilinear pairings, PRFs/OPRFs, are similarly absent. It might be useful to include a table, denoting different key types and algorithms with known efficient threshold solutions. | E34, F4 | – **NOTE:** The report covers only a few examples. Benchmarking may be useful ahead.  
– **CHANGED:** No direct change, but see mention to benchmarking in the new last paragraph in Sec. 7.2. | — |
| 54 | E35 | | Along these lines, there was no discussion of the relationship between the shares generated of a key, and the cryptosystem in question. For example, how should the prime Q be selected when using NIST P-256 vs. secp521r1. When different, this means shares used for EC group should not be employed if the prime Q differs between the two cryptosystems. | E35 | – **NOTE:** The characteristics of threshold schemes may depend on characteristics (e.g., cryptographic parameters) of the underlying cryptographic scheme.  
– **CHANGED:** No change. | — |
<table>
<thead>
<tr>
<th>#</th>
<th>Ref</th>
<th>Old location</th>
<th>E: Comments by Christian Cachin; Hugo Krawczyk; Tal Rabin; Jason Resch; Chrysoula Stathakopoulou (IBM)</th>
<th>Related</th>
<th>Reply Notes</th>
<th>Rev</th>
</tr>
</thead>
<tbody>
<tr>
<td>55</td>
<td>E36</td>
<td></td>
<td>It might also be prudent for the document to suggest that the threshold number of servers should restrict &quot;key usages&quot; for each of the shares they hold, e.g., to prevent shares of a signing key from being used in a decryption operation.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>56</td>
<td>E37</td>
<td></td>
<td>An important, but insufficiently discussed topic was a detailed consideration of crash and compromise recovery procedures. For example, after a compromise a share refresh is often not sufficient. A node may have used a private key to identify itself to other shareholders, and if this key is compromised it needs to be replaced. Whether this involves a CA, or a trusted administrative intervention, or manual updates to each of the N-1 other shareholders is an important distinction between various implementations.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>57</td>
<td>E38</td>
<td></td>
<td>The related problem of recovery is also important and extends beyond share reconstruction: for example in the replicated-state-machine sense, it might require replaying all previously broadcast messages on the shared channel to re-synchronize the state of the recovered shareholder. Alternatively, it might require receiving a representation of the current state signed (or otherwise attested to) by at least F+1 shareholders.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>58</td>
<td>E39</td>
<td>Line 1216</td>
<td>While there is mention of distributed key generation and &quot;dealer free&quot; setups, we felt insufficient time was dedicated to this topic. For example, the paragraph at line 1216 discusses dealer-based setup at length but perhaps should also cover distributed key generation protocols (For example: Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, Tal Rabin: Secure Distributed Key Generation for Discrete-Log Based Cryptosystems. J. Cryptology 20(1): 51-83 (2007))</td>
<td>E39, F4</td>
<td></td>
<td>R43</td>
</tr>
<tr>
<td>59</td>
<td>E40</td>
<td></td>
<td>While the document does discuss loss of availability, there was little discussion of denial of service, and what special precautions or advantages exist in threshold cryptography systems.</td>
<td>E40, J12–J14</td>
<td></td>
<td>R57</td>
</tr>
<tr>
<td>60</td>
<td>E41</td>
<td></td>
<td>(additional feedback items are included in the attached document: &quot;Additional-Comments-NIST-8214.rtf&quot;)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

**Note:** Guidelines for key usage are relevant also for threshold schemes.
**Changed:** No change.

**Note:** Aspects of maintenance (Sec. 5.4) are relevant for threshold schemes, including with respect to the type of rejuvenation (Sec. 5.4.2)
**Changed:** No change.

**Note:** The report also considers rejuvenation of stateful nodes: stateless vs. stateful rejuvenation modes (old line 1239); rejuvenation of stateful nodes may require specifying a sub-protocol (old lines 1248–1250).
**Changed:** No change.

**Note:** Distributed Key generation is a relevant topic. For example, Sec. 3.1 cites distributed key-generation protocols for RSA-based and discrete-log based schemes [Ped91, BF97].
**Changed:** In Sec. 3.1.4, added Ref. [GJKR99] that reviews some aspects of [Ped91].

**Note:** Threshold schemes also provide tradeoffs for protection of availability.
**Changed:** In Sec. 4.1.3, added paragraph about attacks on availability.

**Note:** See items E42–E72.
**Changed:** No change.
<table>
<thead>
<tr>
<th>#</th>
<th>Ref</th>
<th>Old location</th>
<th>E: Comments by Christian Cachin; Hugo Krawczyk; Tal Rabin; Jason Resch; Chrysoula Stathakopoulou (IBM)</th>
<th>Related</th>
<th>Reply Notes</th>
<th>Rev</th>
</tr>
</thead>
<tbody>
<tr>
<td>61</td>
<td>E42</td>
<td>Lines 106-107</td>
<td>&quot;This is possible to achieve for NIST-approved algorithms, such as RSA and DSA signatures, and AES enciphering and deciphering.&quot;  &gt; A citation to the work regarding thresholdized AES encryption and decryption would be instructive. For example: <a href="https://eprint.iacr.org/2018/727.pdf">https://eprint.iacr.org/2018/727.pdf</a></td>
<td>E56, J2</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>62</td>
<td>E43</td>
<td>Lines 122-123</td>
<td>&quot;the corruption of a single share (or of a computation dependent on it) may affect the integrity of the output&quot;  &gt; The particular means of verification of partial results can vary quite substantially from scheme to scheme, and depends largely on the cryptographic algorithm and function being computed. Some discussion on these various mechanisms, or citations providing some examples would be useful. Sometimes individual components can be efficiently checked (e.g. as a RSA signature verify operation), other times it may be easier to check the consistency of the final result first, and only fall back to individual verification if that top-level check fails.</td>
<td>K2–K6</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>63</td>
<td>E44</td>
<td>Lines 122-123</td>
<td>&gt; In one example threshold schemes support a transparent verification via an &quot;interactive proof&quot;, (See section 4 of <a href="https://eprint.iacr.org/2018/733.pdf">https://eprint.iacr.org/2018/733.pdf</a>), in another example a client can use a &quot;BLS signature&quot; to prove each server (or the overall combined value) is consistent, at least when operating over a pairing-friendly curve.</td>
<td>E43</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>64</td>
<td>E45</td>
<td>Line 332,333</td>
<td>&quot;scheme has a “3-out-of-3” property.&quot;  &gt; This is a redefinition from the previously given definition of an &quot;f-out-of-n&quot; system where f is the maximum number of tolerated faults. When only numerals are provided, there is no clarity between whether the text is describing a k-out-of-n system or an f-out-of-n system. It might be better to standardize on a single convention.</td>
<td>A2, E45, E60, F2, K11, K12</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>

---

Table of public comments and replies related to Draft NISTIR 8214
<table>
<thead>
<tr>
<th>#</th>
<th>Ref</th>
<th>Old location</th>
<th>Related</th>
<th>Reply Notes</th>
<th>Rev</th>
</tr>
</thead>
</table>
| 65 | E46  | Line 354     | N6      | ~ **NOTE:** Linear independence follows from requiring that any subset of k hyperplanes (each of dimension k−1) in the k-dimensional space intersect in a single point. Besides linear independence, it is also necessary to require that all planes are not orthogonal to the axis of projection (e.g., x₁) used to derive the secret from the point P.  
~ **CHANGED:** Replaced the incorrect requirement of non-zero coefficients \( h_i \), by a new requirement (see item N6). | R20 |
|    |      |              | E47 E51 | ~ **NOTE:** In old line 357, the indicated inequality is a simple statement of a condition for the purpose of a secret sharing scheme. Sec. 5.1 makes additional considerations about parameters \( k, f \) and \( n \). Optimal choices may vary with the threshold scheme, the attack model, and cost considerations.  
~ **CHANGED:** No change. |      |
| 66 | E47  | Line 357     | E47, E51| ~ **NOTE:** Old lines 334-336 (currently the third paragraph of Sec. 2.2) already describe the intended capability (\( k \)-out-of-\( n \) secret-sharing) achieved by both schemes: “any \( k \) parties together can recover a secret shared across \( n \) parties, but \( k−1 \) parties together do not know anything about the secret.”  
~ **CHANGED:** No change, but added paragraph titles to separate better the description of the two schemes. |      |
| 67 | E48  | Line 385     |         |             | R23 |
| 68 | E49  | Line 369     |         | ~ **NOTE:** Agreed.  
~ **CHANGED:** As suggested. |      |

**Table of public comments and replies related to Draft NISTIR 8214**
<table>
<thead>
<tr>
<th>#</th>
<th>Ref</th>
<th>Old location</th>
<th>E: Comments by Christian Cachin; Hugo Krawczyk; Tal Rabin; Jason Resch; Chrysoula Stathakopoulou (IBM)</th>
<th>Related</th>
<th>Reply Notes</th>
<th>Rev</th>
</tr>
</thead>
<tbody>
<tr>
<td>69</td>
<td>E50</td>
<td>Lines 371-372</td>
<td>&quot;This means that the scheme can in practice be used to share very small secrets (e.g., only a few bits),&quot; &gt; &quot;Only a few bits&quot; seems unjustified. Computers have little difficulty operating over fields defined by primes of hundreds or thousands of bits, which is much larger than &quot;only a few&quot;. Other language besides &quot;few bits&quot; and &quot;very small secrets&quot; might be better, for example &quot;secrets on the order of a few hundred bytes or less&quot;. The real limitation therefore is imposed by the inefficiency of having to store N times as much data as the size of the secret, rather than the size of the finite field in which the math is applied.</td>
<td>E47, E51, E54</td>
<td>~ Note: Changes in the threshold set of nodes is a relevant matter and may require its own sub-protocols. ~ Changed: In Sec. 3.1.3, the new paragraph about re-sharing mentions the possibility of changing the threshold structure, including changing the number of parties.</td>
<td>R25</td>
</tr>
<tr>
<td>70</td>
<td>E51</td>
<td>Line 383</td>
<td>&quot;Secret resharing&quot; &gt; In addition to resharing: there are the related concepts of adding shareholders, removing shareholders, replacing shareholders, changing N, changing K, reconstructing lost shares, and rekeying compromised shareholders.</td>
<td>E47, E51, E54</td>
<td>~ Note: Sec. 2.3 is building up on the concept of secret sharing; other rejuvenation aspects are considered in Sec. 5.4.2. ~ Changed: No change.</td>
<td>R44</td>
</tr>
<tr>
<td>71</td>
<td>E52</td>
<td>Line 388</td>
<td>&gt; Beyond resharing, there exists the necessary steps to patch/secure/reboot/reset/rekey systems in when the need for rekeying is based on a detected compromise.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>72</td>
<td>E53</td>
<td>Line 391</td>
<td>&gt; While correct for this example, it might be worth stating that generally, the protocol would re-randomize all coordinates except for the one that represents the secret. As written it sounds as if only one of the coordinates needs to be re-randomized, but really it is (K-1) of the coordinates (or coefficients in Shamir)</td>
<td>N7, E53</td>
<td>~ Note: The paragraph in question was scoped to $k = 2$. ~ Changed: Added short related sentence in Sec. 2.3.</td>
<td>R31</td>
</tr>
<tr>
<td>73</td>
<td>E54</td>
<td>Line 416</td>
<td>&gt; As described this is describing the operation in the case of a central dealer resharing the same secret. But there are substantial differences when resharing in a dealerless protocol. If the details of this are not covered, it may be worth mention that protocols exist to perform resharing in a dealerless configuration.</td>
<td>E54, F3</td>
<td>~ Note: Agreed. ~ Changed: See reply to item F3.</td>
<td>R44</td>
</tr>
<tr>
<td>#</td>
<td>Ref</td>
<td>Old location</td>
<td>E: Comments by Christian Cachin; Hugo Krawczyk; Tal Rabin; Jason Resch; Chrysoula Stathakopoulou (IBM)</td>
<td>Related</td>
<td>Reply Notes</td>
<td>Rev</td>
</tr>
<tr>
<td>---</td>
<td>-----</td>
<td>--------------</td>
<td>-------------------------------------------------</td>
<td>---------</td>
<td>-------------</td>
<td>-----</td>
</tr>
<tr>
<td>74</td>
<td>E55</td>
<td>Line 437,438</td>
<td>It seems appropriate to add [BGW] to the citation of Yao and GMW. These works provide approaches with different performance tradeoffs and applicability. As an example, recent performance optimizations of Yao’s garbled circuit technique lead to protocols that are relatively efficient in computation but heavy in computation. This presents an example where the LAN/WAN distinction may be crucial in choosing the best implementation. For the LAN setting, Yao may be best, but over WAN one would choose BGW (or GMW).</td>
<td>E55, I11</td>
<td>– NOTE: SMPC can be achieved in a variety of settings. – CHANGED: Added early references to unconditionally secure SMPC.</td>
<td>R32</td>
</tr>
<tr>
<td>75</td>
<td>E56</td>
<td>Line 448</td>
<td>It might be more correct to say that all threshold schemes are special cases of SMPC.</td>
<td>E56, J2</td>
<td>– NOTE: We consider SMPC under the typical framework of ideal functionalities and protocols that emulate an ideal functionality. Some systems may internally implement a threshold approach (e.g., secret-sharing, replication) while not being conceptualized in an SMPC manner. – CHANGED: Adjustments to section 2.4, clarifying the intended distinction.</td>
<td>R33, R34, R35</td>
</tr>
<tr>
<td>76</td>
<td>E57</td>
<td>Line 464</td>
<td>On the subject of side-channel attacks, some consideration for designing implementations and algorithms for threshold cryptography (exponentiation on shares, share generation, etc.) should be designed for constant-time operation. Otherwise information about the share could be leaked over time.</td>
<td>A5, E57</td>
<td>– NOTE: In old lines 618–626 we already covered the relevance of constant-time operations. – CHANGED: No change.</td>
<td>R45</td>
</tr>
<tr>
<td>77</td>
<td>E58</td>
<td></td>
<td>Periodic proactive resharing may be an effective countermeasure, especially if applied after so many operations involving a particular share.</td>
<td></td>
<td>– NOTE: Proactive resharing is mentioned across the report. – CHANGED: No change.</td>
<td>—</td>
</tr>
<tr>
<td>78</td>
<td>E59</td>
<td>Lines 676-685</td>
<td>Thresholdizing a key before performing a computation may yield side-channel attack benefits even in a single computation performed on one processor. That is, by randomizing the secret information on shares upon which the computation is performed. Similar to how some implementations mask timing information in RSA by incorporating a random factor which gets factored out.</td>
<td>E59, M7</td>
<td>– NOTE: Several threshold approaches are of interest, including for the single-device setting. Sec. 2.5 mentions some approaches applicable to address side-channel attacks in the single-device setting. – CHANGED: No change.</td>
<td>—</td>
</tr>
<tr>
<td>#</td>
<td>Ref</td>
<td>Old location</td>
<td>E: Comments by Christian Cachin; Hugo Krawczyk; Tal Rabin; Jason Resch; Chrysoula Stathakopoulou (IBM)</td>
<td>Related</td>
<td>Reply Notes</td>
<td>Rev</td>
</tr>
<tr>
<td>----</td>
<td>------</td>
<td>--------------</td>
<td>---------------------------------------------------------------------------------------------------</td>
<td>--------</td>
<td>-------------</td>
<td>-----</td>
</tr>
</tbody>
</table>
| 79 | E60  | Lines 708-710 | > In general f and k may make poor choices for describing threshold schemes, if only because f and k are so commonly used to represent other things. "t" for threshold might be better. The use of subscripts for indicating what kind of threshold is involved is also a good strategy. Perhaps "f" could instead be t, such that "t_s" (the degree of the polynomial) is the threshold of maintaining secrecy, and t_s+1 is the threshold for exposing the secret (t_e), as an example. | A2, E45, E60, F2, K11, K12 | - **NOTE:** We had exemplified the use of subscripts when differentiating thresholds across different security properties. When focusing on a property, e.g., confidentiality (C), identified in subscript, we still use a different symbol (f vs. k) to distinguish the type of threshold (compromise threshold f vs. operational requirement of needed number of participating parties k).  
- **CHANGED:** We made several clarifications in the document. See reply to A2. | R3, R13, R17, R18, R49, R74, R72 |
| 80 | E61  | Line 710    | > There could perhaps be a little more clarification regarding F_i 0 vs. infinity. Perhaps an example would help. | A3,E60,E61 | - **NOTE:** Agreed.  
- **CHANGED:** Changed $f_I = \infty$ to $f_I = n$ and improved explanation. | R52 |
| 81 | E62  | Line 897    | > Another possibility when using PKI is for full threshold security (no single points of compromise), one might consider having N different CAs, each one responsible and recognized as the only CA for issuing certificates of a given node’s shareholder index. | Diversity levels: D7 (AC), E16 (OS), E62 (CA), E71 (ref), E72 (vendors). | - **NOTE:** It is important to consider diversity levels.  
- **CHANGED:** Added example “CA” in the new row “diversity levels” of Table 2. | R71 |
| 82 | E63  | Line 959    | > This is potentially an overloading of k. It is conceivable that a system could have a different recovery threshold for the secret sharing scheme than the (n - f) good nodes that the byzantine fault tolerant communication channel requires. | Diversity levels: D7 (AC), E16 (OS), E62 (CA), E71 (ref), E72 (vendors). | - **NOTE:** The meaning of “good” (resp. “bad”) components depends on what it applies to. A threshold scheme for a cryptographic primitive may be based on a secret-sharing scheme requiring $f + 1$ good shares to reconstruct a secret, whereas the overall threshold scheme may require that $2f$ nodes remain uncompromised, e.g., possibly due to inability to distinguish between delayed vs. faulty nodes. When need-be the thresholds can be distinguished with additional indices.  
- **CHANGED:** No change. | — |
| 83 | E64  | Line 960    | > A non-crashed simple-majority is not sufficient yet (for consensus, because of FLP85 but the protocols need additional timing assumptions to ensure liveness). | Diversity levels: D7 (AC), E16 (OS), E62 (CA), E71 (ref), E72 (vendors). | - **NOTE:** Old line 953 did mention (with citation [FLP85]) that the problem is unsolvable in a deterministic and asynchronous setting. The statement in old line 960 intended to retain the scope of access to randomness and timing assumptions (synchronous communication) mentioned a few lines before (old lines 956–958).  
- **CHANGED:** In Sec. 4.3.4, clarified statement by rementioning the timing assumptions. | R65 |
<table>
<thead>
<tr>
<th>#</th>
<th>Ref</th>
<th>Old location</th>
<th>E: Comments by Christian Cachin; Hugo Krawczyk; Tal Rabin; Jason Resch; Chrysoula Stathakopoulou (IBM)</th>
<th>Related</th>
<th>Reply Notes</th>
<th>Rev</th>
</tr>
</thead>
<tbody>
<tr>
<td>84</td>
<td>E65</td>
<td>Page 23, Table 2</td>
<td>&gt; Consider adding next to the TLS entry &quot;self&quot; (meaning the protocol itself handles the security of the communications, e.g., via direct encryption under public keys).</td>
<td>E65, I14, K10</td>
<td>– <strong>NOTE:</strong> Channel security can be substantiated in various manners. – <strong>CHANGED:</strong> Added examples to the corresponding row in Table 2.</td>
<td>R68</td>
</tr>
<tr>
<td>85</td>
<td>E66</td>
<td>Line 981</td>
<td>&gt; There is a further notion of <em>dual</em> threshold schemes. They introduce a third parameter L (in the paper it is k), which says from how many nodes a contribution is needed to reconstruct. This can be larger than f; for example L = 2*f + 1. See: Christian Cachin, Klaus Kursawe, Victor Shoup: Random Oracles in Constantinople: Practical Asynchronous Byzantine Agreement Using Cryptography. J. Cryptology 18(3): 219-246 (2005)</td>
<td>– <strong>NOTE:</strong> The possibility that k is larger than f + 1 is allowed by our notation. This is common when n &gt;= 3f + 1. – <strong>CHANGED:</strong> No change.</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>86</td>
<td>E67</td>
<td>Line 1034,1035</td>
<td>&gt; Zero Knowledge Proofs (and interactive proofs) can be used not only for verifying correctness of operations and partial contributions to the output of a cryptographic function, but are also critically important to some distributed key generation and resharing protocols. Aside from such proofs, some functionalities are self-verifiable (e.g signatures). Another possibility is using pairings and BLS (when specific pairing friendly curves are used), with the advantage that there is no overhead for the server to produce a proof.</td>
<td>– <strong>NOTE:</strong> ZKPs have wide applicability. The paragraph ended with an example where ZKPs could be useful for the matter discussed in this paragraph. This paragraph does not focus on concrete schemes. – <strong>CHANGED:</strong> No change.</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>87</td>
<td>E68</td>
<td>Line 1038</td>
<td>&gt; It would be appropriate to cite also the paper on Byzantine quorum systems that predates HM00: Dahlia Malkhi, Michael K. Reiter: Byzantine Quorum Systems. Distributed Computing 11(4): 203-213 (1998)</td>
<td>– <strong>NOTE:</strong> Various works discuss notions mentioned in this report. We have two references focused on arbitrary access structures (one from 1989). – <strong>CHANGED:</strong> No change.</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>88</td>
<td>E69</td>
<td>Line 1079</td>
<td>&quot;Are clients aware of the threshold nature of the implementation?&quot; &gt; Perhaps this property can be referred to as &quot;transparency&quot; and highlighted as an important feature in many settings. The definition of transparency being that the user of the service cannot distinguish between a threshold implementation and a centralized service. (This is mentioned section 5.2 but the transparency advantage is not highlighted). There is also the question of whether ZK proofs of correctness can be verified transparently (e.g., combined from each of the threshold server’s individual ZK or interactive proofs by a service aggregator or proxy).</td>
<td>E69, E70, I12</td>
<td>– <strong>NOTE:</strong> The property suggestively called “transparency” also conveys “opacity”, since it equates to hiding the threshold structure from the client. The property can be relevant; in some cases the opposite is also an advantage, e.g., in multi-signatures. Sec. 5.2 is about communication interfaces, but the suggested property is not only about communication interfaces. The question about ZK may be specific to concrete threshold schemes. See complementary observation in item I12. – <strong>CHANGED:</strong> No change.</td>
<td>—</td>
</tr>
<tr>
<td>#</td>
<td>Ref</td>
<td>Old location</td>
<td>Related</td>
<td>Reply Notes</td>
<td>Rev</td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td></td>
</tr>
</tbody>
</table>
| 89 | E70 | Line 1261 | E69, E70, I12 | ~ **NOTE:** It can still satisfy the notion of $f$ for some property, e.g., key confidentiality if the key is shared across all nodes. While non-exposure of the primary may be advantageous in some settings, there are conceivable settings where it may be advantageous to have it exposed. For example, a setting that requires a full audit-trail of the threshold process could want to identify who was the leader during a threshold execution.  
~ **CHANGED:** Slight adjustment in sentence about communication via primary node. | R77 |
| 90 | E71 | Line 1270 | Diversity levels: D7 (AC), E16 (OS), E62 (CA), E71 (ref), E72 (vendors). | ~ **NOTE:** It is important to consider diversity levels.  
~ **CHANGED:** Added row “diversity levels” to Table 2; added suggested reference that surveys the use of diversity. | R71, R88 |
| 91 | E72 | Line 1291 | Diversity levels: D7 (AC), E16 (OS), E62 (CA), E71 (ref), E72 (vendors). | ~ **NOTE:** It is important to consider diversity levels.  
~ **CHANGED:** In Sec. 5, added new row “diversity level” in Table 2, including examples “vendors” as an example. | R71 |
<table>
<thead>
<tr>
<th>#</th>
<th>Ref</th>
<th>Old location</th>
<th>F: Comments by Tanja Lange (TUE)</th>
<th>Related</th>
<th>Reply Notes</th>
<th>Rev</th>
</tr>
</thead>
</table>
| 92 | F1  |              | I was surprised that the draft is a meta document and apart from the basic Blakey and Shamir schemes does not include any details. The questions posted are reasonable and relevant when evaluating schemes. I understand that making selections will make the document much more controversial but I think it would also make it more useful. Covering everything from basic secret sharing schemes to applications of MPC in side-channel protection is a daunting task and I wonder what the future of this document is meant to be – are there going to be many appendices covering the actual schemes, organized by application or by technology used? | D1, E1, F1, G1, H2, H8, I1, J1, K1, L1, M2 | ~ **NOTE:** The last paragraph of Sec. 1 explains our decision to not cover the most recent research results. This document intends to motivate an initial engagement with/from stakeholders. We hope to hear from stakeholders, e.g., during NTCW’19, about comparative advantages of various threshold schemes. We do not envision updating this document with appendices, but future documents in the scope of this project may cover more concrete technical material.  
~ **CHANGED:** No change. | —  |
| 93 | F2  |              | Editorial comment: It is a bit confusing that intro and section 2 talk about f-out-of-n threshold schemes and section 1 talks about k-out-of-n secret sharing schemes. A short version of section 5.1 given in the introduction would help to clarify that there are two different directions. It might also help to highlight this in the examples, so mention 1-out-of-2 threshold or basic Blakey etc. I agree that both are common but this can be confusing. | A2, E45, E60, F2, K11, K12 | ~ **NOTE:** Agree with needed clarifications.  
~ **CHANGED:** See reply to A2. | R3, R13, R17, R18, R49, R74, R72 |
| 94 | F3  | Section 2.2  | Content comment: I think it’s a missed chance to not mention that resharin for Shamir works without ever reconstructing the secret, so also without the need for a dealer. I would like to see a short section on this after line 418. Ensuring that devices securely forget data is a problem in practice and an important reason for using secret-sharing schemes is the requirement not to trust a single party, so there should never be a party that knows the secret – for generation or for resharin or for usage. If you would rather include a reference than a worked out section, may I suggest my work with Yvo Desmedt https://link.springer.com/chapter/10.1007%2F11889663_12 in which we go through the full details of what this means on the example of ID-based cryptography. | E54, F3 | ~ **NOTE:** Resharin without a dealer is a relevant capability.  
~ **CHANGED:** Added a short paragraph in Sec. 3.1.3, mentioning the possibility of resharin without a dealer. | R44 |
<table>
<thead>
<tr>
<th>#</th>
<th>Ref</th>
<th>Old location</th>
<th>F: Comments by Tanja Lange (TUE)</th>
<th>Related</th>
<th>Reply Notes</th>
<th>Rev</th>
</tr>
</thead>
</table>
| 95 | F4  | Section 3.1  | Content comment: The example of RSA signatures is nice in that it shows both the applicability and the restrictions. However, it would also be good to include a fully positive example, such as ElGamal decryption or DH key share computation where there group order is known and thus k-out-of-n schemes can be used in a distributed manner (as opposed to n-out-of-n). | E34, E35, E39, F4 | – **NOTE:** In old line 549 there was already a citation that achieves k-out-of-n even for an RSA-based scheme. We appreciate that the use of groups of known order can sometimes facilitate some properties in threshold schemes (e.g., as mentioned in old lines 568-572). Old lines 570–579 also alluded to easier operations when the group order is known.  
– **CHANGED:** We intend to avoid intricate details, but, to emphasize the suggested point, made some edits in the “k-out-of-n threshold scheme” paragraph: mentioned the aspect of knowledge vs. lack-of-knowledge of group order; cited [DF90] as complementary example; also new citation [GJKR99] in Sec. 3.1.3. | R38 |
<table>
<thead>
<tr>
<th>#</th>
<th>Ref</th>
<th>Old location</th>
<th>Related</th>
<th>Reply Notes</th>
<th>Rev</th>
</tr>
</thead>
</table>
| 96 | G1  | G: Comments by Yehuda Lindell (Bar-Ilan and Unboundtech) | D1, E1, F1, G1, H2, H8, I1, J1, K1, L1, M2 | ~ **NOTE:** Thank you for the encouragement.  
~ **CHANGED:** No change. | — |
| 97 | G2  | Secure multiparty computation (MPC) has reached a level of maturity that makes it suitable for solving the problem of key protection via key distribution, without the key ever being reassembled. There are fast protocols for all of the standard cryptographic algorithms (RSA key generation, signing and decryption, DH/ECDH, IES/ECIES, ECDSA/DSA, AES with all modes of operation, HMAC and so on), and there is considerable interest from industry in deploying these solutions. As such, we now see a number of startups and larger companies deploying MPC for this purpose. We have three main comments, that we outline below. | Industry: B1, G2, I2, M1; Schemes: E25, E27, G2. | ~ **NOTE:** This NISTIR intends to promote discussion, including with industry, about threshold schemes for cryptographic primitives.  
~ **CHANGED:** No change. | — |
| 98 | G3  | **FIPS certification.** The purpose of standardization of protocols in this area must be to enable FIPS certification at a higher level than currently now possible. Otherwise, it is not clear what concrete advantage one would obtain by having these standards. It is currently possible to achieve FIPS 140-2 level 1 and level 2 using MPC and threshold schemes (by having the different modules connected via FIPS 140-2 certified TLS channels), where level 2 additionally requires certified OS and hardware. However, FIPS 140-2 levels 3 and 4 are currently reserved for dedicated HSMs only. | G3, I7, M6 | ~ **NOTE:** The aspect of validation/certification is important, and the development of new standards and the various nuances of characterizing features may involve reconsidering validation requirements. In old lines 1477–1478, in Sec. 8, we mention that the “fit into FIPS 140-2 ... is a matter for discussion”. Old representative question #18 (new #6a) asks to check whether security assertions match FIPS 140.  
~ **CHANGED:** In Sec. 6.2, added new paragraph mentioning that “the process towards standardization of threshold schemes may involve reconsidering the adequacy of the validation requirements and where necessary devise new of complementary requirements”. | R89 |
<table>
<thead>
<tr>
<th>#</th>
<th>Ref</th>
<th>Old location</th>
<th>G: Comments by Yehuda Lindell (Bar-Ilan and Unboundtech)</th>
<th>Related</th>
<th>Reply Notes</th>
<th>Rev</th>
</tr>
</thead>
<tbody>
<tr>
<td>99</td>
<td>G4</td>
<td></td>
<td>We believe that the threat model achieved via MPC and threshold cryptography is different but comparable to that of dedicated HSMs, with each having advantages and disadvantages. For example, MPC solutions can be compromised if a quorum of participants are breached, unlike HSMs. However, any flaw inside an HSM is extremely expensive and difficult to recover from, increasing the expected loss. Due to the inflexibility of hardware, this also means that HSMs often do not support the best latest standards and take a long time to deploy new standards. In addition, the main threat that is considered in levels 3 and 4 beyond lower levels is due to attacks requiring physical access. In today’s cloud environments, such physical attacks can be greatly mitigated using other means. Finally, in the modern computing era with cloud computing, HSMs in the cloud lose a lot of their security benefits (primarily since they are no longer under the organization’s physical control). As such, one should have alternatives in threshold schemes and MPC that provide level 3 and 4 security, and can be deployed in such scenarios.</td>
<td>G4, E14–E16.</td>
<td>– NOTE: There are relevant differences between HSMs vs. cloud nodes. – CHANGED: See reply to HSM-related items E14–E16.</td>
<td>R7</td>
</tr>
<tr>
<td>100</td>
<td>G5</td>
<td></td>
<td><strong>What to standardize.</strong> It appears to be natural to standardize popular tools, like Shamir secret Sharing, Yao’s garbled circuits, oblivious transfer, and so on. We support standardizing these, but warn against these being the only thing certified. This is due to the fact that constructing a secure MPC protocol involves far more than the use of good primitives. It is true that certified encryption schemes can also be misused, but with MPC it is much harder to do this correctly.</td>
<td>G5, I6, J7, M3, M4</td>
<td>– NOTE: Composability is an important matter to consider. – CHANGED: In Sec. 7.2, added a new paragraph highlighting the caution needed with composability of modular components. The new text in Sec. 4.1.4 mentions composability as one aspect to consider in proofs. The new representative question 5e asks about composability.</td>
<td>R100, R108</td>
</tr>
<tr>
<td>#</td>
<td>Ref</td>
<td>Old location</td>
<td>G: Comments by Yehuda Lindell (Bar-Ilan and Unboundtech)</td>
<td>Related</td>
<td>Reply Notes</td>
<td>Rev</td>
</tr>
<tr>
<td>----</td>
<td>-----</td>
<td>--------------</td>
<td>-----------------------------------------------------</td>
<td>--------</td>
<td>-------------</td>
<td>-----</td>
</tr>
</tbody>
</table>
| 101 | G6  |              | To make this situation worse, the rate of advancement of MPC is so great that any particular protocol that is standardized will most likely be out of date very quickly. As such, we propose that a specific suite of protocols for all of the standard cryptographic algorithms be standardized (including RSA, DH/ECDH, DSA/ECDSA, and symmetric algorithms), but it be explicitly stated that improvements on these protocols also be allowed. This raises a problem as to how a different protocol can be certified. We argue that publication in a recognized journal for peer review (like the Journal of Cryptology) and/or external validation via a recognized MPC researcher at a recognized institution, should be valid. We are fully aware that defining such requirements is a huge minefield. As such, we don’t have an answer as to how this should be done, but we do strongly argue that a way must be found. Otherwise, the standardization process may yield little benefit. | G6, J5 | – **NOTE:** The process of standardizing new threshold schemes will consider the “NIST cryptographic standards and guidelines development process” (NISTIR 7977) [Gro16]. Useful experience will be gained throughout the process of standardization.  
– **CHANGED:** In Sec. 4.1.4 (“proofs of security”) added a reference to the “NIST Cryptographic Standards and Guidelines Development Process” (NISTIR 7977) [Gro16], which establishes principles that NIST follows for the development of crypto standards. | R59 |
| 102 | G7  |              | **Proofs of security.** In general, proofs of security are now well accepted as necessary for cryptographic schemes. For MPC, they are absolutely essential. We argue that both simulation-based and game-based definitions should be acceptable, but that security should always be required in the presence of malicious adversaries with a reasonable security parameter (e.g., computational parameter at the level of the algorithm, and statistical parameter at least $2^{-40}$). There are some cases where weaker levels of security in the presence of malicious adversaries, like that provided by dual execution, can also be considered. These issues need to be clarified and discussed as the tradeoffs are important. | G7, I38, J8, J9, J10, H6 | – **NOTE:** We agree that proofs are nowadays essential.  
– **CHANGED:** In Sec. 4.1.4 (“proofs of security”), mentioned several aspects to consider in proofs, including types of proof and security parameters. In Sec. 7.1, replaced old question #14 with more focused questions (5e, 5f, 5g) that ask about properties of the proof of security. | R58, R100, R101, R102 |
<table>
<thead>
<tr>
<th>#</th>
<th>Ref</th>
<th>Old location</th>
<th>H: Comments by Samuel Ranellucci (Unboundtech)</th>
<th>Related</th>
<th>Reply Notes</th>
<th>Rev</th>
</tr>
</thead>
</table>
| 103 | H1  |              | To summarize our comments, we believe that having different types of standards would result in simpler standards that are easier to understand. | H1, H3, H9 | – **NOTE:** See reply to item H3  
– **CHANGED:** No change. | – |
| 104 | H2  |              | We would like to thank NIST for taking the initiative on standardizing threshold implementations. | D1, E1, F1, G1, H2, H8, I1, J1, K1, L1, M2 | – **NOTE:** Thank you for the encouragement.  
– **CHANGED:** No change. | – |
| 105 | H3  |              | We believe that NIST could make it easier to standardize threshold implementations by creating different types of specifications. We begin by describing a problem that could be solved by creating different types of specifications. | H1, H3, H9 | – **NOTE:** The aspect of granularity of standards (Sec. 7.2) is a main open question.  
– **CHANGED:** No change. | – |
| 106 | H4  |              | To prove a threshold implementation secure, it is necessary to define many features. The list of features includes but is not limited to the necessity of a trusted setup, access to some hybrid functionalities, access to cryptographic primitives, the existence of private and authenticated channels between parties, and includes the power of the adversary (semi-honest or malicious, static or adaptive). Already, these issues would make constructing a specification complex. However, if we also include issues such as “What levels of diversity are envisioned to deter common-mode failures?” and “Is the system model applicable to known and relevant application contexts?”, this would make standardization even more complicated. To remedy this problem, we recommend three types of specifications: primitive standardization, protocol standardization and application standardization. |                              | – **NOTE:** The report enumerates several of these many features. Levels of diversity are now further exemplified in Table 2. Composability is an important consideration when considering composition of different layers of specifications.  
– **CHANGED:** No change. | – |
| 107 | H5  |              | **Primitive Specifications:** The first type of specifications are implementations of cryptographic primitives in the same vein as AES and SHA2 standardization. Standardizing garbling schemes and secret sharing fall under this category. | H5, I31, J6 | – **NOTE:** Relates to representative question 2c. The aspect of granularity of standards (Sec. 7.2) is a main open question.  
– **CHANGED:** New paragraph in Sec. 7.2 about modular components. See also the reply to items about composability (G5, I6, J7, M3, M4). | R108 |

Table of public comments and replies related to Draft NISTIR 8214
Protocol Specifications: The second type of specifications would be directed at interactive protocols. For these specifications, the model should not contain system information and should only contain the model typically found in cryptographic papers. In addition, I would recommend that unless stated otherwise, these specifications should assume private and authenticated channels, static security and malicious adversaries since this is the most relevant model in the literature. GMW-based protocols and garbled-circuit protocols fall under this category.

Deployment Specifications: Finally, in the last type of specification, the specifications focus on system issues that are crucial to the deployment of threshold implementations. These specifications, when built upon the second type specifications, could be described at a higher level of abstraction. Threshold implementations of private-data storage and distributed password verification would fall under this category. We are aware that for certain applications, precise deployment specifications could not be generated due to the variance of requirements. However, it may be the case that for certain applications, standardization would be beneficial.

In conclusion, we thank NIST for their effort.

We believe that our recommendation of having different types of standards would result in simpler standards that are easier to understand.
Ref | Old location | Comments by Dan Bogdanov (Cybernetica) | Related | Reply Notes | Rev
---|---|---|---|---|---
112 | I1 | Cybernetica is happy to see that threshold cryptography is gaining momentum and best practices are being collected by NIST. We see this as a critical step for the new technologies to be accepted in the United States of America. | D1, E1, F1, G1, H2, H8, I1, J1, K1, L1, M2 | – **Note:** Thank you for the encouragement. – **Changed:** No change. | —
113 | I2 | We have multiple US partners who are evaluating secure multi-party computation for data analytics uses. | B1, G2, I2, M1(industry), C1, I2, I5, E30, H7 (higher-level apps) | – **Note:** This NISTIR intends to promote discussion, with multiple stakeholders, about threshold schemes for cryptographic. SMPC is a main approach to achieve threshold schemes. – **Changed:** No change, but see item I5. | —
114 | I3 | This document consists of suggestions to further development in this field and responses to a subset of questions. Please find the responses to questions in separate chapters | | – **Note:** We appreciate the comments/answers provided. Many of these questions are intended to be addressed in the context of concrete schemes, i.e., what should be questioned/considered when analyzing concrete schemes, or when pondering about how to ask for proposals for new schemes. – **Changed:** No change. | —
115 | I4 | **Suggestion S1:** If the aim is to set the scope on using threshold cryptography for cryptographic algorithms (e.g., key generation, encryption, decryption, signatures), then be as clear as possible about this. Threshold cryptography is a general-purpose technique and can be used for big data, statistics etc applications. If NIST standardises the use of these techniques for encryption and signature use, make sure that the same restrictions may not be reasonable for data analytics use. | I5 | – **Note:** See reply to item I5. – **Changed:** No change. | —
116 | I5 | **Suggestion S2:** If the aim is to set the scope on using threshold cryptography for a wide range of applications (e.g., key generation, encryption, decryption, signatures, but also data general purpose processing), then allow these applications to have separate characteristics. | C1, I2, I5, E30, H7 (higher-level apps) | – **Note:** This report is focused on threshold schemes for “cryptographic primitives”. There are higher-level applications that can take advantage of such schemes. – **Changed:** Added a brief note on the motivation for higher-level applications, in a place that is immediately followed by a paragraph that conveys the focus of this report: cryptographic primitives. | —
117 | I6 | **Suggestion S3:** If the scope will be on wider applications, avoid standardizing monolithic protocols. Support and encourage composition of primitives. | G5, I6, J7, M3, M4 | – **Note:** See reply to item I5. – **Changed:** About composability, see reply to G5. | —

Table of public comments and replies related to Draft NISTIR 8214

89
<table>
<thead>
<tr>
<th>#</th>
<th>Ref</th>
<th>Old location</th>
<th>I: Comments by Dan Bogdanov (Cybernetica)</th>
<th>Related</th>
<th>Reply Notes</th>
<th>Rev</th>
</tr>
</thead>
</table>
| 118 | I7  |              | **Suggestion S4:** Avoid limitations to certain security levels. E.g., some applications may be provably secure in the honest-but-curious security model, but will protect against other attacks using other approaches. | G3, H6, I7, L2, L3 | – **NOTE:** Evaluation of potential threshold schemes and their relations with security levels is a relevant matter.  
– **CHANGED:** See replies to G3 (on security levels) and L2 (on different types of adversary). | R66, R89 |
| 119 | I8  |              | **Suggestion S5:** Encourage machine-generated and machine-checked protocol implementations. | E11, I8 | – **NOTE:** Relates to automated validation.  
– **CHANGED:** See item E11. | R91, R96 |
| 120 | I9  | m            | Section 5.1, lines 1040-1041 3 QUESTIONS ON THRESHOLD VALUES 3.1 QUESTION 1: FOR THE DESIRED SECURITY PROPERTIES, WHAT ARE THE THRESHOLD VALUES \( F \) AND/OR \( K \), AS A FUNCTION OF THE TOTAL NUMBER \( N \) OF COMPONENTS?  
It is not clear if this question assumes that all parties in the system (including customers) are involved in \( n \). If yes, then the threshold may be a function of \( n \). But if some users of the system are not part of \( n \) (e.g., they delegate their computation to the parties holding the shares), then it does not have to be a function. It may take a few shares per input (2, 3 even) to process inputs from millions of users. This highly depends on the application, as raising the number of components often reduces performance. We recommend no hard restrictions here. | – **NOTE:** When the threshold system is conceived as cryptographic module (to perform some cryptographic primitive), the “customers” (users, requesters of cryptographic operations) are typically not included in \( n \). More elaborate answers may vary with the system model.  
– **CHANGED:** No change. | — |
| 121 | I10 | Section 5.1, lines 1042-1043 3.2 QUESTION 2: WHAT ENVISIONED APPLICATION CONTEXTS JUSTIFY A HIGH THRESHOLD FOR SOME PROPERTIES AT THE COST OF A LOW THRESHOLD FOR OTHER PROPERTIES (OR OF OTHER MITIGATION MEASURES)?  
In a way, the threshold gives us a slider between control-confidentiality and integrity-reliability. Low thresholds ensure that even if some parties misbehave by sending wrong results or not sending anything at all, the results can still be calculated. High thresholds are better for confidentiality. We recommend no hard restrictions here. | – **NOTE:** There are various possible tradeoffs.  
– **CHANGED:** No change. | — |
<table>
<thead>
<tr>
<th>#</th>
<th>Ref</th>
<th>Old location</th>
<th>Comments by Dan Bogdanov (Cybernetica)</th>
<th>Related</th>
<th>Reply Notes</th>
<th>Rev</th>
</tr>
</thead>
</table>
| 122| I11 | Section 5.1, lines 1044-1045 | 3.3 QUESTION 3: HOW DO THRESHOLD VALUES VARY WITH RESPECT TO CONCEIVABLE VARIATIONS OF THE SYSTEM MODEL (E.G., SYNCHRONY VS. ASYNCHRONY, PASSIVE VS. ACTIVE ADVERSARIES)? There are well-known possibility results, several of them cited in Draft NISTIR 8214 (CCD87 etc). However, we are not aware of a function that takes as input the system model and outputs a recommended number of parties. We see that the number of components and thresholds are often application-specific. We recommend no hard restrictions here. | E55, I11 | – NOTE: Thresholds may vary with the system model.  
– CHANGED: No direct change but [CCD88] was added. | R32 |
| 123| I12 | Section 5.2, line 1079 | 4 QUESTIONS ON COMMUNICATION INTERFACES 4.1 QUESTION 1: ARE CLIENTS AWARE OF THE THRESHOLD NATURE OF THE IMPLEMENTATION? If the human users are not, then the client interfaces (libraries, websites, mobile apps, adapters or whatever) have to be. Pushing the actual secret sharing process to the client gives better auditability and security in all constructions that we are aware of. We recommend no hard restrictions here. | E69, I12 | – NOTE: Auditability may be a feature of certain threshold schemes, e.g., for multi-signatures, as mentioned in Sec. 3.1.4. For certain applications there may be advantages in having the threshold interface not reveal the threshold internals. See complementary observation in item E69.  
– CHANGED: No change. | — |
| 124| I13 | Section 5.2, line 1080 | 4.2 QUESTION 2: HOW IS THE INITIAL REQUEST FROM A CLIENT PROPAGATED THROUGH THE SET OF NODES? This is highly application-specific. In some cases, encrypting shares at the customers and propagating them to the shareholders through a single gateway is preferable, sometimes it is better to send shares directly to the parties that will process them. We recommend no hard restrictions here. | E65, I14, K10 | – NOTE: The preference can depend on the application and also on the attack model.  
– CHANGED: No change. | — |
| 125| I14 | Section 5.2, line 1081 | 4.3 QUESTION 3: HOW CAN THE INTER-NODE COMMUNICATION BE COMPROMISED? Given that threshold cryptography implementations are often modern, they have a better probability to use up-to-date secure channel techniques. In practice, we see that compromises are more probable at the endpoints. | E65, I14, K10 | – NOTE: Communication channels may vary substantially, e.g., with the implementation platform.  
– CHANGED: No change. Added more examples to the corresponding row in Table 2. | R68 |
<table>
<thead>
<tr>
<th>#</th>
<th>Ref</th>
<th>Old location</th>
<th>Comments by Dan Bogdanov (Cybernetica)</th>
<th>Related</th>
<th>Reply Notes</th>
<th>Rev</th>
</tr>
</thead>
<tbody>
<tr>
<td>126</td>
<td>I15</td>
<td>Section 5.2, lines 1082-1083</td>
<td>4.4 QUESTION 4: HOW DOES THE CLIENT OBTAIN A CONSOLIDATED REPLY BASED ON A SET OF PARTIAL RESULTS PRODUCED BY A SET OF NODES? This is application-specific. However, in most applications, we see that the client often has the best information to decide on the results. This is because the client can receive partial results from various committees and several nodes.</td>
<td>– NOTE: It may depend on the application. – CHANGED: No change.</td>
<td></td>
<td>—</td>
</tr>
<tr>
<td>127</td>
<td>I16</td>
<td>Section 5.2, lines 1084-1085</td>
<td>4.5 QUESTION 5: HOW IS THE LOGICAL/PHYSICAL “BOUNDARY” (SEE FIPS 140-2 [NIS18C]) OF THE SYSTEM AFFECTED BY THE EXISTING COMMUNICATION CHANNELS? This is application-specific. In some settings, we consider a set of parties holding the shares for a single value a single logical computer. E.g., in a general-purpose computation scenario based on secure multi-party computation, all parties holding the shares form a single application server.</td>
<td>– NOTE: For validation purposes, answers will also depend on implementation details, prior to deployment for a higher-level application. – CHANGED: No change.</td>
<td></td>
<td>—</td>
</tr>
<tr>
<td>128</td>
<td>I17</td>
<td>Section 5.3, lines 1200-1201</td>
<td>5 QUESTIONS ON TARGET COMPUTING PLATFORMS 5.1 QUESTION 1: IF A PROPOSED THRESHOLD SCHEME IS DEVISED FOR A “SINGLE-DEVICE” SETTING, WHAT CAN GO WRONG IF ITS COMPONENTS ARE INSTEAD SEPARATED AND COMMUNICATE OVER THE INTERNET? This question does not have enough information for a certain response. One could design a scheme that works in a single-device setting and also remains secure in a distributed setting. And, in the case of a protocol based on trusted execution environments, this could not work.</td>
<td>– NOTE: The answer may vary, e.g., depending on the composability guarantees of the communication “module”. – CHANGED: No change.</td>
<td></td>
<td>—</td>
</tr>
<tr>
<td>129</td>
<td>I18</td>
<td>Section 5.3, lines 1202-1203</td>
<td>5.2 QUESTION 2: WHICH PARTS OF THE LOGICAL BOUNDARY OF THE THRESHOLD SYSTEM DO NOT CORRESPOND TO A PHYSICAL BOUNDARY, AS VERIFIED BY THE SYSTEM DEVELOPER OR DEPLOYER? No response at this time.</td>
<td></td>
<td></td>
<td>—</td>
</tr>
<tr>
<td>#</td>
<td>Ref</td>
<td>Old location</td>
<td>I: Comments by Dan Bogdanov (Cybernetica)</td>
<td>Related</td>
<td>Reply Notes</td>
<td>Rev</td>
</tr>
<tr>
<td>----</td>
<td>-----</td>
<td>--------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>---------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-----</td>
</tr>
<tr>
<td>130</td>
<td>I19</td>
<td>Section 5.3, lines 1204-1205</td>
<td>5.3 QUESTION 3: IS THE SYSTEM SIMPLY DEVELOPED AT THE SOFTWARE LAYER, OR ARE THERE SOFTWARE COMPONENTS TIED TO PARTICULAR HARDWARE COMPONENTS? We can imagine both kinds of solutions. Consider threshold protocols where it is important that a certain part of the deployment is in a mobile phone or in a Trusted Execution Environment. We recommend no hard restrictions here.</td>
<td>E17, I19</td>
<td>– NOTE: We can also imagine hybrid solutions. – CHANGED: No change.</td>
<td></td>
</tr>
<tr>
<td>131</td>
<td>I20</td>
<td>Section 5.3, lines 1206-1207</td>
<td>5.4 QUESTION 4: WHICH AUXILIARY COMPONENTS SUPPORT THE THRESHOLD SCHEME BUT HAVE A FAILURE MODEL DIFFERENT FROM THE ONE APPLIED TO THE THRESHOLD NODES? No response at this time.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>132</td>
<td>I21</td>
<td>Section 5.4, lines 1299</td>
<td>6 QUESTIONS ON SETUP AND MAINTENANCE 6.1 QUESTION 1: CAN A THRESHOLD SCHEME BE BOOTSTRAPPED IN BOTH DEALER AND DEALER-FREE MANNERS? We can imagine various deployments. In dealer-free environments, shares could be generated by parties. In single-dealer environments, there will be one authoritative party for shares. In a multi-dealer environment, there will be multiple dealers who all know the scheme and the parties to share the secrets to. Thus, the answer would be yes, and we also foresee many multi-dealer deployments.</td>
<td></td>
<td>– NOTE: We can also envision multiple solutions. An open question is whether a single standard for a threshold scheme should/may incorporate both types of solutions. The multi-dealer observation scenario is interesting. – CHANGED: No change.</td>
<td></td>
</tr>
<tr>
<td>133</td>
<td>I22</td>
<td>Section 5.4, lines 1300</td>
<td>6.2 QUESTION 2: WHAT LEVELS OF DIVERSITY ARE ENVISIONED TO DETER COMMON-MODE FAILURES? No response at this time.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>134</td>
<td>I23</td>
<td>Section 5.4, lines 1301</td>
<td>6.3 QUESTION 3: WHAT DEPENDENCY OF COMPROMISE EXISTS ACROSS NODES, FOR ENVISIONED ATTACK VECTORS? No response at this time.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>135</td>
<td>I24</td>
<td>Section 5.4, lines 1302</td>
<td>6.4 QUESTION 4: DOES THE SUBPROTOCOL FOR HANDLING REJUVENATIONS INTERFERE WITH THE SYSTEM AVAILABILITY? No response at this time.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#</td>
<td>Ref</td>
<td>Old location</td>
<td>I: Comments by Dan Bogdanov (Cybernetica)</td>
<td>Related</td>
<td>Reply Notes</td>
<td>Rev</td>
</tr>
<tr>
<td>----</td>
<td>-----</td>
<td>--------------</td>
<td>------------------------------------------</td>
<td>---------</td>
<td>-------------</td>
<td>-----</td>
</tr>
<tr>
<td>136</td>
<td>I25</td>
<td>Section 7, lines 1374-1377</td>
<td>7 QUESTIONS ON CRITERIA FOR STANDARDIZATION FOR NEW SCHEMES 7.1 QUESTION 1: ARE THE CHARACTERIZING FEATURES OF THE THRESHOLD SCHEME FULLY DESCRIBED? We believe that once the scope for standardization is set (see Suggestions S1 and S2), it will become easier to figure out the required characteristics.</td>
<td>~ <strong>NOTE:</strong> Agreed; here we also wanted to propose that once a new scheme is analyzed one should ask if/which of its characterizing features are well described. ~ <strong>CHANGED:</strong> No change.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>137</td>
<td>I26</td>
<td>Section 7, lines 1378</td>
<td>7.2 QUESTION 2: ON WHAT EXECUTING PLATFORMS CAN THE SCHEME BE IMPLEMENTED? A relevant question, if there are restrictions (e.g., Trusted Execution Environments like ARM TrustZone or Intel SGX).</td>
<td>~ <strong>NOTE:</strong> Different schemes may require different assumptions on trusted components, so some schemes/assumptions might not be possible to implement/cover in some platforms. ~ <strong>CHANGED:</strong> No change.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>138</td>
<td>I27</td>
<td>Section 7, lines 1379</td>
<td>7.3 QUESTION 3: WHAT ARE THE OPERATIONAL COSTS AND PROPERTIES OF SETUP AND MAINTENANCE? Often this will be dependent on third party services, e.g. clouds.</td>
<td>~ <strong>NOTE:</strong> It will be pertinent for a new threshold scheme proposal to estimate related costs. ~ <strong>CHANGED:</strong> No change.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>139</td>
<td>I28</td>
<td>Section 7, lines 1380</td>
<td>7.4 QUESTION 4: WHAT ARE THE NODE-REJUVENATION MECHANISMS (E.G., RESHARING OR NODE REPLACEMENT)? A relevant question.</td>
<td>~ <strong>NOTE:</strong> Agreed. ~ <strong>CHANGED:</strong> No change.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>140</td>
<td>I29</td>
<td>Section 7, lines 1381</td>
<td>7.5 QUESTION 5: HOW EFFICIENT/PERFORMANT ARE THE OPERATIONS AS A FUNCTION OF THRESHOLD PARAMETERS? A relevant question. But note that some schemes might have a fixed n and k, still being threshold schemes.</td>
<td>~ <strong>NOTE:</strong> Even for fixed parameters, it will be relevant to compare costs vs. the conventional (non-threshold) system. When parameters can vary, it will also be interesting to know how the cost varies. ~ <strong>CHANGED:</strong> No change.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>141</td>
<td>I30</td>
<td>Section 7, lines 1382</td>
<td>7.6 QUESTION 6: IS THE SCHEME APPLICABLE TO NIST-APPROVED CRYPTOGRAPHIC PRIMITIVES? A relevant question.</td>
<td>~ <strong>NOTE:</strong> Agreed. ~ <strong>CHANGED:</strong> No change.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>142</td>
<td>I31</td>
<td>Section 7, lines 1383</td>
<td>7.7 QUESTION 7: DO BASE PRIMITIVES (E.G., OBLIVIOUS TRANSFER) REQUIRE INDEPENDENT STANDARDIZATION? A relevant question. However note that this should be informative, as the use of a new useful protocol should not be delayed because a primitive is not standardized.</td>
<td>H5, I31, J6 ~ <strong>NOTE:</strong> This may become relevant if it provides advantages for standardization of higher-level primitives. For now it is an open question. ~ <strong>CHANGED:</strong> No change. See reply to item H5.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>#</td>
<td>Ref</td>
<td>Old location</td>
<td>I: Comments by Dan Bogdanov (Cybernetica)</td>
<td>Related</td>
<td>Reply Notes</td>
<td>Rev</td>
</tr>
<tr>
<td>----</td>
<td>-----</td>
<td>----------------------</td>
<td>----------------------------------------------------------------------------------------------------------</td>
<td>--------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------------</td>
<td>-----</td>
</tr>
</tbody>
</table>
| 143| I32 | Section 7, lines 1384| 7.8 QUESTION 8: IS THE SYSTEM MODEL APPLICABLE TO KNOWN AND RELEVANT APPLICATION Contexts?              |                                                                          | - **NOTE:** Agreed.  
- **CHANGED:** No change.                                                   |     |
| 144| I33 | Section 7, lines 1385| 7.9 QUESTION 9: HOW IS DIVERSITY OF NODES RELATED TO KNOWN ATTACK VECTORS?                             |                                                                          | - **NOTE:** Diversity is relevant for threshold schemes.  
- **CHANGED:** We have added a new line to Table 2, for “diversity levels”. |     |
| 145| I34 | Section 7, lines 1386| 7.10 QUESTION 10: IS THE IMPLEMENTATION COMPLEXITY LIKELY TO LEAD TO NEW BUGS OR MISCONFIGURATION?     |                                                                          | - **NOTE:** The goal of the question is to promote caution on possible new bugs and misconfigurations that may arise from the threshold approach. Somewhat related, Sec. 4.2 also considers “threshold-related” attacks.  
- **CHANGED:** Edited the question to become more open, asking to identify which aspects can lead to new bugs or misconfiguration | R97 |
| 146| I35 | Section 7, lines 1387| 7.11 QUESTION 11: WHAT TRUSTED SETUP AND ASSUMPTIONS ARE REQUIRED (E.G., DEALER, SPECIAL COMPONENTS)? |                                                                          | - **NOTE:** Agreed.  
- **CHANGED:** No change.                                                   |     |
| 147| I36 | Section 7, lines 1388| 7.12 QUESTION 12: WHAT THRESHOLD PROPERTIES RELATE TO RESISTANCE AGAINST SIDE-CHANNEL ATTACKS AND HOW? |                                                                          | - **NOTE:** Side-channels resistance can relate to different parts of a system. Some threshold approaches may mitigate some side-channel attacks.  
- **CHANGED:** No change.                                                   |     |
| 148| I37 | Section 7, lines 1389| 7.13 QUESTION 13: ARE THERE IDENTIFIED SECURITY TRADEOFFS ACROSS ATTACK TYPES AND CONFIGURATIONS?     |                                                                          | - **NOTE:** Agreed.  
- **CHANGED:** No change.                                                   |     |
| 149| I38 | Section 7, lines 1390| 7.14 QUESTION 14: IS THE SECURITY ASSESSMENT SUPPORTED BY A SECURITY PROOF?                           | G7, I38, J8, J9, J10                                                    | - **NOTE:** Agreed.  
- **CHANGED:** See the reply to G7.                                          | R58, R100, R101, R102 |
<table>
<thead>
<tr>
<th>#</th>
<th>Ref</th>
<th>Old location</th>
<th>Comments by Dan Bogdanov (Cybernetica)</th>
<th>Related</th>
<th>Reply Notes</th>
<th>Rev</th>
</tr>
</thead>
</table>
| 150 | I39 | Section 7, lines 1391 | 7.15 QUESTION 15: HOW DOES THE RELIABILITY COMPARE AGAINST THAT OF A CONVENTIONAL IMPLEMENTATION? A question with uncertain relevance. If the threshold scheme is built for confidentiality, it will always be less reliable. If it is built for reliability, it will be more reliable. |         | ~ **NOTE:**  We mean reliability in upholding a desired security property, e.g., including being reliable in upholding the secrecy of a key.  
~ **CHANGED:**  No change. | —   |
| 151 | I40 | Section 7, lines 1392 | 7.16 QUESTION 16: HOW BRITTLE IS THE SCHEME (LIKELY TO BREAK UNDER SMALL VARIATIONS IN THE ENVIRONMENT)? A question with uncertain relevance. Isn’t this a partial duplicate for questions 9, 10, 11? |         | ~ **NOTE:**  Ideally, a good threshold scheme would remain secure under conceivable variations in the environment.  
~ **CHANGED:**  No change. | —   |
| 152 | I41 | Section 7, lines 1393 | 7.17 QUESTION 17: WHAT FEATURES OF GRACEFUL DEGRADATION EXIST AGAINST CONCEIVABLE FAILURES? A relevant question. |         | ~ **NOTE:**  Agreed.  
~ **CHANGED:**  No change. | —   |
| 153 | I42 | Section 7, lines 1394 | 7.18 QUESTION 18: DO THE SECURITY ASSERTIONS MATCH / FIT INTO THE FIPS 140-2 FRAMEWORK? A relevant question. |         | ~ **NOTE:**  Agreed.  
~ **CHANGED:**  No change. | —   |
| 154 | I43 | Section 7, lines 1395 | 7.19 QUESTION 19: HOW TESTABLE IS THE SCHEME (CAN SECURITY ASSERTIONS BE TESTED AND VALIDATED)? A very relevant question. |         | ~ **NOTE:**  Agreed.  
~ **CHANGED:**  No change. | —   |
| 155 | I44 | Section 7, lines 1396 | 7.20 QUESTION 20: IS THERE A PROPOSED AUTOMATED VALIDATION MECHANISM? A relevant question. |         | ~ **NOTE:**  Agreed.  
~ **CHANGED:**  No change. | —   |
| 156 | I45 | Section 7, lines 1397 | 7.21 QUESTION 21: WHAT ARE THE INTELLECTUAL PROPERTY IMPLICATIONS AND THE LICENSING CONDITIONS? A relevant question. |         | ~ **NOTE:**  Agreed.  
~ **CHANGED:**  No change. | —   |
<table>
<thead>
<tr>
<th>#</th>
<th>Ref</th>
<th>Old location</th>
<th>J: Comments by Rosario Gennaro (The City College of New York)</th>
<th>Related</th>
<th>Reply Notes</th>
<th>Rev</th>
</tr>
</thead>
</table>
| 157 | J1  |              | In this short document I present a few comments on the draft NIST document on threshold cryptography. Before I begin, I would like to commend NIST for starting a project of standardization of threshold cryptographic schemes. In the process you are raising the awareness of the security community about the availability of sophisticated cryptographic techniques that can enhance security (intended in the largest possible sense including confidentiality, integrity, availability, etc.). | D1, E1, F1, G1, H2, H8, I1, J1, K1, L1, M2 | – **NOTE:** Thank you for the encouragement.  
– **CHANGED:** No change. | — |
| 158 | J2  |              | **Generic vs Ad-hoc Solutions.** It should be remarked that any cryptographic primitive can be implemented in a distributed "threshold" manner, via generic secure multiparty computation protocols. | E56, J2 | – **NOTE:** Agreed.  
– **CHANGED:** Some adjustments in Sec. 2.4, making this point clear, while also conveying that some threshold techniques can also be conceptualized outside the SMPC framework. | R33, R34 |
| 159 | J3  |              | The dramatic improvement in speed and reliability of some of those protocols of the last few years suggests that maybe we should consider  
(i) generic solutions for cryptographic primitives for which we do not have ad-hoc protocols;  
(ii) a requirement that compares ad-hoc solutions to generic multiparty computation approaches | J3–J5 | – **NOTE:** Generic solutions are a potential possibility.  
– **CHANGED:** See reply to J5. | — |
| 160 | J4  |              | (ii) a requirement that compares ad-hoc solutions to generic multiparty computation approaches | J3–J5 | – **NOTE:** Ad-hoc solutions are a potential possibility.  
– **CHANGED:** See reply to J5. | — |
| 161 | J5  |              | In other words I wonder if this standardization effort should include a standard way to turn any cryptographic primitive into a threshold one using a secure MPC protocol of choice (and standardizing ways to express the primitive into a format that can be fed into the MPC protocol, e.g. how to write it as a circuit, etc.). Once that is done, ad-hoc protocols would have a "standard" benchmark to be compared against the generic approach (and be adopted only if they show substantial improvements over the generic approach). | G6, J3–J5 | – **NOTE:** Might there be synergies to extract between the two types of solutions?  
– **CHANGED:** Added related paragraph in Sec. 7.2. | R109 |
<table>
<thead>
<tr>
<th>#</th>
<th>Ref</th>
<th>Old location</th>
<th>J: Comments by Rosario Gennaro (The City College of New York)</th>
<th>Related</th>
<th>Reply Notes</th>
<th>Rev</th>
</tr>
</thead>
</table>
| 162 | J6  |              | For the ad-hoc solutions I also wonder if we need standard protocols for "building blocks" that appear across many solutions. An incomplete list would include: secret sharing, VSS, interpolation in the "exponent" (i.e. compute $g^x$ from shares of $x$), computing shares of the inverse of a secret value (i.e. go from a share of $x$ to a share of $x^{-1}$ over some group), and so on. | H5, I31, J6 | – **NOTE:** See reply to item I31.  
– **CHANGED:** See reply to item H5. | R108 |
| 163 | J7  |              | If those building blocks are standardized and proven secure according to a composable definition of security, then the design of protocols for primitives could be greatly simplified, and implementation of different cryptographic primitives would share a common "language". | G5, I6, J7, M3, M4 | – **NOTE:** Composability is an important matter to consider.  
– **CHANGED:** See reply to G5. | R108 |
| 164 | J8  |              | **THE NEED FOR SECURITY PROOFS.** The draft document asks at the end (question 14) if security proofs should be required. My answer to that question is an emphatic yes. A security proof should guarantee that moving from a centralized to a distributed implementation of the cryptographic primitive would not introduce other weaknesses beyond the ones of the centralized primitive. Therefore if the centralized primitive is assumed secure then the distributed version would be as well, under such a security proof. | G7, I38, J8, J9, J10 | – **NOTE:** Agreed.  
– **CHANGED:** See reply to item J9. | – |

Table of public comments and replies related to Draft NISTIR 8214
<table>
<thead>
<tr>
<th>#</th>
<th>Ref</th>
<th>Old location</th>
<th>Comments by Rosario Gennaro (The City College of New York)</th>
</tr>
</thead>
<tbody>
<tr>
<td>165</td>
<td>J9</td>
<td>The draft document falls a bit short in my opinion in discussing the options for security proofs. It mentions (somewhat in passing) the real-ideal simulation paradigm which is of course the gold standard for this type of security proofs. However it should be remarked that in some cases proofs based on a &quot;game-based&quot; definitions might suffice. In a game-based definition the salient property of the underlying cryptographic primitive is re-defined in a distributed model where the adversary also has access to the information of the players in the network. For example, if we are distributing a signature scheme, the game-based definition of unforgeability against chosen-message attack can be adapted to the distributed model and the distributed signature proven secure according to this definition. Game-based definitions are weaker since they do not rule out distributed protocols where the adversary might learn information, though they guarantee that this information will not help in breaking the underlying signature. The lack of a generic composability guarantee is the major drawback of this approach.</td>
<td></td>
</tr>
<tr>
<td>166</td>
<td>J10</td>
<td>I suggest that the draft document be improved by describing in more detail the possible choices for security proofs, and the various scenarios in which one proof may be acceptable, and that overall the need for security proofs be stressed as mandatory.</td>
<td></td>
</tr>
<tr>
<td>167</td>
<td>J11</td>
<td>THE NEED FOR AVAILABILITY. The document does an excellent job at discussing the trade-offs between confidentiality and availability when using a threshold scheme: if the operation of the scheme requires all parties to participate, then confidentiality is at its highest, but the scheme will not be robust as a single party may force the system to shut down. On the other hand availability can be trivially improved by replicating the secret keys across servers, but this makes confidentiality weaker, as now the key is more exposed to attacks. The problem, as discussed in the draft document, is to find the correct &quot;equilibrium point&quot; which may be application dependent.</td>
<td></td>
</tr>
</tbody>
</table>

**Related**

G7, I38, J8, J9, J10

**Reply Notes**

- **NOTE:** Agreed.
- **CHANGED:** In Sec. 4.1, added a new paragraph on “Proofs of security”. Also replaced the previous representative question #14 with three new more focused questions (5e, 5f, 5g) about security proofs.

R58, R100, R101, R102
<table>
<thead>
<tr>
<th>#</th>
<th>Ref</th>
<th>Old location</th>
<th>J: Comments by Rosario Gennaro (The City College of New York)</th>
<th>Related</th>
<th>Reply Notes</th>
<th>Rev</th>
</tr>
</thead>
</table>
| 168 | J12 | | Availability can always be improved by having a sufficiently high number of servers participate in the protocol, so that even if \( f \) servers do not participate there is enough honest servers to maintain functionality of the system. However for efficiency reason one may want to run the protocol with the minimum possible number of servers and hope in an "optimistic" fashion that the servers are honest. But what happens if things go wrong. | E40, J12–J14 | - \textbf{NOTE:} See item J14  
- \textbf{CHANGED:} Added related paragraph in Sec. 4.1.3. | R57 |
| 169 | J13 | | One aspect that might be important to discuss in this context, is the ability to detect and reverse faults. For example if \( k \) servers are needed to compute the primitive and then the protocol fail because \( f \) of those servers might be faulty. The questions here are | J13–J17 | - \textbf{NOTE:} Detection of faults and failure is an important matter.  
- \textbf{CHANGED:} Added representative question 4e about fault detection. | R99 |
| 170 | J14 | | - If \( k \) is sufficiently high can we still successfully compute the primitive | J13–J17 | - \textbf{NOTE:} Detectability vs. non-detectability of faults may change the threshold numbers, depending on the type of attack.  
- \textbf{CHANGED:} Added related paragraph in Sec. 4.1.3. | R57 |
| 171 | J15 | | - If not, can we at least detect failure (this is easy with signatures as the resulting signature will not verify – but what about other primitives, e.g. PRFs)? | J13–J17, K6 | - \textbf{NOTE:} Different primitives may pose different challenges for correctness verification (e.g., see reply to K6).  
- \textbf{CHANGED:} No change. | — |
| 172 | J16 | | - If we detect failure, can we identify the rogue server(s) and remove them from the server pool replacing them with other servers? | J13–J17, K9 | - \textbf{NOTE:} The “removal/replacement” aspect relates to representative question 1c. The “identification” part is handled by the reply to J13.  
- \textbf{CHANGED:} See reply to J13. | — |
| 173 | J17 | | These features could be important to have in certain applications. | J13–J17 | - \textbf{NOTE:} Applications may benefit from properties of threshold schemes.  
- \textbf{CHANGED:} No change. | — |
<table>
<thead>
<tr>
<th>#</th>
<th>Ref</th>
<th>Old location</th>
<th>K: Comments by Thalia May Laing (HP Inc.)</th>
<th>Related</th>
<th>Reply Notes</th>
</tr>
</thead>
</table>
| 174| K1  |              | We thought the draft document covered some main points related to secret sharing schemes and found it to raise some interesting and important questions. We have the following suggestions as to how the document could be improved and some concepts clarified. | D1, E1, F1, G1, H2, H8, I1, J1, K1, L1, M2 | ~ NOTE: Thank you for the encouragement.  
~ CHANGED: No change. |
<p>| 175| K2  |              | (1) Verifiable secret sharing schemes are the main family of schemes recommended to provide robustness and they do so, whilst assuming the dealer is untrusted. Robust secret sharing schemes provide robustness in a more efficient manner whilst assuming the dealer is trusted yet are not recommended as a family of schemes. Is there a requirement for an untrusted dealer? Is there a reason why verifiable secret sharing are suggested and robust schemes are not? | Robustness: K2–K6 | ~ NOTE: See reply to item K5. |
| 176| K3  |              | In a verifiable secret sharing scheme (VSS), the dealer is not assumed to be trusted. When a party receives a share from the dealer, they can verify that the share they received is a valid share. Then, when each player submits their share to recover the secret during the reconstruct phase, each share can also be verified to ensure that dishonest players cannot completely disrupt the recovery process. In a robust secret sharing scheme, the dealer is assumed to be trusted and the aim is to prevent dishonest players from corrupting the recovery process by submitting incorrect shares. As we assume the dealer to be trusted we assume the players are initially dealt correct shares. | Robustness: K2–K6 | ~ NOTE: See reply to item K5. |
| 177| K4  | Lines 128, 1033 | Despite this, robust schemes are not suggested as a family of schemes to provide robustness and, instead, verifiable schemes are recommended. For example, in line 128, it is said that 'verifiable secret sharing enables detection of misuse of shares by a shareholder', and on line 1033, verifiable secret sharing schemes are suggested as the solution to a threshold number of parties misbehaving. This is correct, as verifiable secret sharing schemes achieve this, but they also assume an untrusted dealer. Robust schemes do this in a more efficient way (multiple rounds of communication are not necessary) but assuming the dealer to be trusted. | Robustness: K2–K6 | ~ NOTE: See reply to item K5. |</p>
<table>
<thead>
<tr>
<th>#</th>
<th>Ref</th>
<th>Old location</th>
<th>K: Comments by Thalia May Laing (HP Inc.)</th>
<th>Related</th>
<th>Reply Notes</th>
<th>Rev</th>
</tr>
</thead>
<tbody>
<tr>
<td>178</td>
<td>K5</td>
<td>It may be helpful to the reader to mention robust schemes as another suggested family of schemes that provide robustness (as well as VSS) and highlight the difference between the trust assumptions on the dealer and the necessary rounds of communication. Examples of robust secret sharing schemes can be found in Section 4 of Krawczyk’s 'Secret sharing made short’ paper, and in Sections 4 and 5 of Bellare and Rogaway’s unified account of secret sharing goals. Krawczyk, H. (1993, August). Secret sharing made short. In Annual International Cryptology Conference (pp. 136-146). Springer, Berlin, Heidelberg. Bellare, M., &amp; Rogaway, P. (2007, October). Robust computational secret sharing and a unified account of classical secret-sharing goals. In Proceedings of the 14th ACM conference on Computer and communications security (pp. 172-184). ACM.</td>
<td>Robustness: K2–K6.</td>
<td>– NOTE: Both cases (trusted vs. untrusted dealers) are interesting to consider, as well as the case without dealer. – CHANGED: In Sec. 5.4.1, added mention to the distinction between the two cases: trusted vs. untrusted dealer.</td>
<td>R83</td>
<td></td>
</tr>
<tr>
<td>179</td>
<td>K6</td>
<td>(2) It may be useful to the reader to highlight both the efficiency costs and the security requirements on threshold schemes with additional properties (such as robust, verifiable and proactive schemes). Some schemes may have extra communication and complexity costs: - Robust schemes require the communication of more bits from the dealer to the players. - Verifiable secret sharing requires the communication of more bits and multiple rounds of communication. For example, the [AMGC85], a scheme highlighted in the document, is an interactive protocol with multiple rounds of communication. This is an increased communication cost over standard threshold schemes. - In proactive secret sharing schemes, players may have to generate random values and may need to send every other player a value, which is an increased computation and communication cost.</td>
<td>Robustness: J15, K2–K6.</td>
<td>– NOTE: The efficiency of threshold schemes is a relevant matter. As an example, old lines 1029-1031 (new Sec. 5.1.3) mention that verifying the correctness of encryption or decryption may be costlier (see Sec. 5.1.3). – CHANGED: See also the reply to item K5.</td>
<td>R84</td>
<td></td>
</tr>
<tr>
<td>180</td>
<td>K7</td>
<td>As well as efficiency costs, the schemes may have additional security requirements:</td>
<td></td>
<td></td>
<td>C</td>
<td></td>
</tr>
<tr>
<td>181</td>
<td>K8</td>
<td>- Some VSS constructions may require security properties on channels between parties. For example, [Fel87] (a scheme referenced in the document) assumes a private channel from each player to every other player.</td>
<td>Channels: H6, K8</td>
<td>– NOTE: We mention that schemes can have different properties based on the system model, including the communication model. – CHANGED: No change.</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>#</td>
<td>Ref</td>
<td>Old location</td>
<td>K: Comments by Thalia May Laing (HP Inc.)</td>
<td>Related</td>
<td>Reply Notes</td>
<td>Rev</td>
</tr>
<tr>
<td>----</td>
<td>-----</td>
<td>--------------</td>
<td>------------------------------------------</td>
<td>---------</td>
<td>-------------</td>
<td>-----</td>
</tr>
</tbody>
</table>
| 182 | K9  |              | - Proactive schemes may require some method of recognising whether a party has been corrupted and a way to 'reset' a corrupted party, so it is no longer corrupted, | J16, K9 | ~ **NOTE**: Both “reactive” and “proactive” settings are relevant, with respect to “recognizing” corruptions and initiating rejuvenation of components.  
~ **CHANGED**: No change. | — |
| 183 | K10 |              | More generally, it may also be useful to highlight that the dealer must securely delegate shares of the secret to the players in all settings. | E65, I14, K10 | ~ **NOTE**: The ability to handle untrusted dealers may also be relevant.  
~ **CHANGED**: No change, but see items E65 and I14. | — |
| 184 | K11 | 247-248      | (3) It may improve the reader’s understanding if the notation is consistent throughout. Specifically, k has two meanings throughout the document, which can lead to confusion. In Section 1, an ‘f out of n’ scheme is referred to (lines 247 and 248), meaning that the scheme is resilient to up to f out of n parties being compromised. In Sections 2.1 and 2.2, it is a ‘k out of n’ scheme (used in the sense that k-1 parties together do not know anything about the secret) and then in Section 2.5 the ‘f’ notation is used, while Section 3.1 is back to k. At this early stage in the document, is it true that f = k-1? Would it make sense to use one symbol (such as just k, and replace f with k-1) until it is necessary to split them? | A2, E45, E60, F2, K11, K12 | ~ **NOTE**: We agree that a clarification is useful.  
~ **CHANGED**: See reply to A2. | R3, R13, R17, R18, R49, R74, R72 |
| 185 | K12 | 995-996      | In Section 5.1 (lines 982-983), f is again used to denote the number of bad components and k is the minimum required number of good components. This use of k is conflicting with the notation used earlier in the document and could lead to confusion. In lines 995-996, it is said that in an n out of n signature scheme, f=n-1 while k=1. If f is the number of parties that can be compromised then it is clear that f=n-1 and, using your more recent definition of k, it is clear that k=1. But if k is used according to the original definition in the document, then k=n in this example as n players are required to compute a signature. Harmonising the notation throughout will help make the text clearer. | A2, E45, E60, F2, K11, K12 | ~ **NOTE**: This set of paragraphs shows how f and k depend on the security property at stake. It purposely shows how the same scheme can have different values k for different properties.  
~ **CHANGED**: Moved the explanation of possible/omitted indices (C, A, I) to earlier in section 5.1 (now within section 5.1.1) to make it more clear upfront. Also added indices in more uses of f and k. | R73, R74 |
<table>
<thead>
<tr>
<th>#</th>
<th>Ref</th>
<th>Old location</th>
<th>L: Comments by Karim Eldefrawy (SRI International)</th>
<th>Related</th>
<th>Reply Notes</th>
<th>Rev</th>
</tr>
</thead>
<tbody>
<tr>
<td>186</td>
<td>L1</td>
<td>The NISTIR 8214 document has done a great job covering most aspects involved in threshold cryptographic schemes, e.g., adversary and system model, dynamic groups, connectivity between devices, proactive security guarantees.</td>
<td>D1, E1, F1, G1, H2, H8, I1, J1, K1, L1, M2</td>
<td>– <strong>NOTE:</strong> Thank you for the encouragement. – <strong>CHANGED:</strong> No change.</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>187</td>
<td>L2</td>
<td>An additional issue to consider is that of mixed adversaries. Most existing proactive secret sharing (PSS) schemes only guarantee secrecy in the presence of an honest majority with at most ( n/2 - 1 ) total corruptions during such a refresh period; an adversary that corrupts a single additional party beyond the ( n/2 - 1 ) threshold, even if only passively and only temporarily, obtains the secret.</td>
<td>H6, I7, L2, L3</td>
<td>– <strong>NOTE:</strong> The aspect of graceful degradation mentioned in Sec. 4.3.4 can also be considered for the case where the compromise threshold is exceeded. – <strong>CHANGED:</strong> See reply to item L3.</td>
<td>R66</td>
<td></td>
</tr>
<tr>
<td>188</td>
<td>L3</td>
<td>Recent work [DELOY16] developed a PSS scheme secure in the presence of a dishonest majority. The PSS scheme is robust and secure against ( t &lt; n-2 ) passive adversaries when there are no active corruptions, and secure but non-robust (but with identifiable aborts) against ( t &lt; n/2-1 ) active adversaries when there are no additional passive corruptions. The scheme is also secure (with identifiable aborts) against mixed adversaries controlling a combination of passively and actively corrupted parties such that if there are ( k ) active corruptions there are less than ( n-k-1 ) total corruptions. The PSS scheme was then utilized to demonstrate feasibility of Proactive Secure Multiparty Computation (PMPC) with a dishonest majority with similar security guarantees. [DELOY16] Dolev S., Eldefrawy K., Lampkins J., Ostrovsky R., Yung M. (2016) Proactive Secret Sharing with a Dishonest Majority. In: Zikas V., De Prisco R. (eds) Security and Cryptography for Networks. SCN 2016. Lecture Notes in Computer Science, vol 9841. <a href="https://link.springer.com/chapter/10.1007/978-3-319-44618-9_28">https://link.springer.com/chapter/10.1007/978-3-319-44618-9_28</a></td>
<td>H6, I7, L2, L3</td>
<td>– <strong>NOTE:</strong> It is pertinent to compare how a protocol behaves under different types of attackers (passive vs. active), and what kind of (graceful?) degradation may exist when a threshold of compromise is surpassed. – <strong>CHANGED:</strong> Added statement about dual thresholds in the end of Sec. 4.3.4. See also the reply to item L4.</td>
<td>R66</td>
<td></td>
</tr>
<tr>
<td>#</td>
<td>Ref</td>
<td>Old location</td>
<td>L: Comments by Karim Eldefrawy (SRI International)</td>
<td>Related</td>
<td>Reply Notes</td>
<td>Rev</td>
</tr>
<tr>
<td>----</td>
<td>-----</td>
<td>---------------</td>
<td>--------------------------------------------------</td>
<td>---------</td>
<td>-------------</td>
<td>-----</td>
</tr>
</tbody>
</table>
| 189 | L4  |               | Ongoing work is improving the communication and computation complexity of such PMPC scheme and by the time of the 2019 NIST workshop on threshold cryptography, more efficient PMPC schemes for dishonest majority should be publicly available. We plan to submit a workshop talk proposal overviewing the above issue, and these PSS and MPC protocols and modifications thereof that improve their communication and computation complexity and render them much more practical. [EOPY18] Eldefrawy K., Ostrovsky R., Park S., Yung M. (2018) Proactive Secure Multiparty Computation with a Dishonest Majority. In: Catalano D., De Prisco R. (eds) Security and Cryptography for Networks. SCN 2018. Lecture Notes in Computer Science, vol 11035. https://link.springer.com/chapter/10.1007/978-3-319-98113-0_11 | – NOTE: We welcome submissions to NTCW’19.  
– CHANGED: No change. | — |
<table>
<thead>
<tr>
<th>#</th>
<th>Ref</th>
<th>Old location</th>
<th>M: Comments by John Wallrabenstein (Analog)</th>
<th>Related</th>
<th>Reply Notes</th>
</tr>
</thead>
</table>
| 190 | M1 | A substantial barrier to the productization and adoption of novel cryptographic constructions is the requirement from customers to comply with existing NIST standards, particularly FIPS 140-2. While the list of standardized algorithms covers the core components of cryptographic systems, it remains a small subset of the cryptographic primitives discussed in peer-reviewed literature. This limitation beneficially ensures that algorithms are only adopted after careful cryptanalysis. However, many well-studied cryptographic primitives (e.g., threshold schemes, zero knowledge proofs, etc.) remain unstandardized and are therefore onerous to integrate into products targeting heavily regulated markets despite their ability to address known adversarial attack strategies. | B1, G2, I2, M1 | – **NOTE:** the present document intends to promote standardization of threshold schemes  
– **CHANGED:** No change | — |
| 191 | M2 | Analog Devices, Inc. supports the goal of Draft NISTIR 8214 to standardize threshold cryptographic schemes, which provide resiliency against a wide variety of real-world adversarial attack strategies. | D1, E1, F1, G1, H2, H8, I1, J1, K1, L1, M2 | – **NOTE:** Thank you for the encouragement.  
– **CHANGED:** No change. | — |
| 192 | M3 | * On the granularity of certification, we suggest certifying individual primitives (e.g., threshold ECIES, distributed key generation, etc.) rather than larger composed constructions. This provides implementers a high degree of freedom in tailoring threshold systems to their specific application, and reduces the burden on the standard itself. | G5, I6, J7, M3, M4 | – **NOTE:** Composability is an important matter to consider; we intend to promote standardization of threshold schemes for diverse cryptographic primitives.  
– **CHANGED:** In Sec. 7.2, a new **paragraph** highlights the caution needed with composability of modular components. | R108 |
| 193 | M4 | The existing FIPS 140-2 standard already places the burden of ensuring primitives are combined in a secure way onto the implementer, and we suggest the same approach for a threshold standard. | G5, I6, J7, M3, M4 | – **NOTE:** Somewhat unclear what is the suggested approach — let implementers combine primitives into a threshold scheme, or let implementers combine threshold schemes into higher-level applications? Composability is an important matter to consider.  
– **CHANGED:** See reply to M3. | R108 |
| 194 | M5 | * It may be useful to first focus on standardizing threshold implementations of RSA, elliptic curve cryptography, and AES for the single device setting. The communication, consensus, and automated validation requirements for the multi-device setting appear to introduce far more obstacles to standardization than the single device setting, which removes many of these issues. |  | – **NOTE:** We have identified both single-device and multiparty settings as part of the scope of promoting threshold schemes for cryptographic primitives. Different obstacles in different settings may be tackled in parallel.  
– **CHANGED:** No change. | — |
<table>
<thead>
<tr>
<th>#</th>
<th>Ref</th>
<th>Old location</th>
<th><strong>M</strong>: Comments by John Wallraabenstein (Analog)</th>
<th>Related</th>
<th><strong>Reply Notes</strong></th>
<th>Rev</th>
</tr>
</thead>
</table>
| 195 | M6  |              | * A standard covering threshold versions of existing standard-ized cryptographic primitives in the single device setting also allows easier integration with existing NIST test harnesses for their corresponding non-threshold versions. | G3, M6 | **NOTE:** Test and validation procedures may possibly come to depend on the characterizing features of threshold schemes  
**CHANGED:** In section 6.2, added text mentioning that “the process towards standardization of threshold schemes may involve reconsidering the adequacy of the validation requirements and where necessary devise new of complementary requirements”. | R89 |
| 196 | M7  | Lines 577-579 | * Distributed key generation was briefly mentioned (lines 577-579, Pedersen '91) in the draft, although the context appears to consider only the multi-device setting. We suggest that in addition to the multi-device setting, distributed key generation for the single device setting (where all parties in the “distributed” key generation protocol reside on a single device) be standardized. | E59, M7 | **NOTE:** Distributed key generation is relevant for both single-device and multiparty settings. See reply to E59.  
**CHANGED:** No change. | |
| 197 | M8  |              | * Is there a tentative timeline and roadmap for arriving at a NIST standard(s) for threshold cryptographic schemes? | | **NOTE:** This NISTIR was an initial step, to be followed by the NIST Threshold Cryptography Workshop 2019 (March 11–12). A tentative timeline and roadmap does not yet exist. We welcome feedback from stakeholders.  
**CHANGED:** No change. | C |
<table>
<thead>
<tr>
<th>#</th>
<th>Ref</th>
<th>Old location</th>
<th>N: Comments by authors (NIST) — editorial</th>
<th>Related</th>
<th>Reply Notes</th>
<th>Rev</th>
</tr>
</thead>
<tbody>
<tr>
<td>198</td>
<td>N1</td>
<td>Abstract, line 50</td>
<td>Clarify upfront that threshold schemes are composed of multiple components that contribute to the intended outcome.</td>
<td></td>
<td>– <strong>CHANGED:</strong> Adjust sentence that mentions components for the first time (R1).</td>
<td></td>
</tr>
<tr>
<td>199</td>
<td>N2</td>
<td>Line 1029</td>
<td>Add acknowledgments for public comments.</td>
<td></td>
<td>– <strong>CHANGED:</strong> Added acknowledgments to the contributors of public comments (R2).</td>
<td></td>
</tr>
<tr>
<td>200</td>
<td>N3</td>
<td></td>
<td>Various editorial revisions: avoid contractions (apostrophes), correct typos, commas, define all acronyms, ...</td>
<td></td>
<td>– <strong>CHANGED:</strong> apostrophe contractions (R37,R39,R48,R90); commas (R46); missing verb (R87); citation tags as words (R78, R79); others (R85); “down time” (R86); RSA and AES (R4); CPU, SGX, ARM (R81); HSM (R80); RNG (R70).</td>
<td></td>
</tr>
<tr>
<td>201</td>
<td>N4</td>
<td>Section 7.1</td>
<td>Reorganize the set of representative questions, considering their increased number.</td>
<td></td>
<td>– <strong>CHANGED:</strong> Organized the representative questions by topic; the indexing of questions is now made with a lower case letter (topic) and number (R92).</td>
<td></td>
</tr>
<tr>
<td>202</td>
<td>N5</td>
<td>Line 1104, 1115</td>
<td>For better referencing, some portions of text can be promoted to numbered subsections or subsubsections.</td>
<td>E20</td>
<td>– <strong>CHANGED:</strong> Several subsections of Sections 3, 4 and 5 now have numbered subsubsections; Section 7 now has numbered subsections.</td>
<td></td>
</tr>
<tr>
<td>203</td>
<td>N6</td>
<td>Line 1029</td>
<td>Blakley scheme: shares in 2-out-of-(n) Blakley must be non-vertical; shares in (k)-out-of-(n) must be non-orthogonal to the (x_1) axis.</td>
<td></td>
<td>– <strong>CHANGED:</strong> Made explicit the orthogonality requirement: R27, R28, R30, R21 Added sentence selecting (x_1) as the coordinate for the secret (R22).</td>
<td></td>
</tr>
<tr>
<td>204</td>
<td>N7</td>
<td>Line 356</td>
<td>For the (k)-out-of-(n) Blakley scheme with (k &gt; 2), explain how to extract the secret from the intersection point (P).</td>
<td>N7, E53</td>
<td>– <strong>CHANGED:</strong> Added sentence selecting (x_1) as the coordinate (of (P)) that defines the secret (R22).</td>
<td></td>
</tr>
<tr>
<td>205</td>
<td>N8</td>
<td></td>
<td>Some relocations across paragraphs.</td>
<td></td>
<td>– <strong>CHANGED:</strong> Old subsection “2.5 Terminology” is now Sec. 2.1, at the beginning of Section “Fundamentals” (R16); note on ISO/IEC secret-sharing is moved up a few paragraphs, to the end of new Sec. 2.2 (R26); In Sec. 2.2, note on (n) vs. (Q) is moved up one paragraph R19. Old 2nd paragraph of section 8 (conclusions) moved to (and then adjusted in) 1st paragraph of new Sec. 7.3 (R110).</td>
<td></td>
</tr>
<tr>
<td>#</td>
<td>Ref</td>
<td>Old location</td>
<td>N: Comments by authors (NIST) — editorial</td>
<td>Related</td>
<td>Reply Notes</td>
<td>Rev</td>
</tr>
<tr>
<td>---</td>
<td>-----</td>
<td>--------------</td>
<td>------------------------------------------</td>
<td>---------</td>
<td>-------------</td>
<td>-----</td>
</tr>
<tr>
<td>206</td>
<td>N9</td>
<td></td>
<td>Some text adjustments.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>– <strong>CHANGED:</strong> cite more recent attack (Foreshadow) (R9); “key” → “secret” (R25); cite previous descriptions of threshold RSA and mention “dealer” (R36); “universal” → “global” (R42); in section 4, old line 736, clarify [low thresholds] “be dealt with at a different application layer” (R55); replace “meta-questions” by “questions about ...” (R106); replace “centralized authority” by “same entity” (R63); various other text improvements (R6, R8, R10, R14, R35, R40, R47, R50, R75, R76).</td>
<td></td>
</tr>
<tr>
<td>207</td>
<td>N10</td>
<td>After line 760</td>
<td>Connect the importance of security models / pitfalls of decoupling security properties to the case of threshold schemes.</td>
<td></td>
<td>– <strong>CHANGED:</strong> Added corresponding paragraph (R56).</td>
<td></td>
</tr>
</tbody>
</table>