

“Preview Writeup”: In anticipation of a package submission to the NIST Threshold Call

Title: Kryptallage

Subtitle: Thresholdized FHE scheme switching

Version: Preview Writeup 1.0 (2026-09-25)¹

Team name: Kryptallage team

Team members: Ghada Almashaqbeh, Sonia Belaïd, Nicolas Bon, Ilaria Chillotti, Mariana Gama, Antonio Guimarães, Erin Hales, Hyojun Kim, Elena Kirshanova, Damien Ligier, Ryan Orendorff, Emmanuela Orsini, Matthieu Rivain, Luisa Siniscalchi, Ravital Solomon, Rick Weber

Abstract: We present a threshold integrated FHE system that unifies several homomorphic encryption schemes and the procedures for switching between them into a single encrypted-computation framework. Rather than organizing around named schemes (CGGI, CKKS, BGV, BFV), we treat the ciphertext type—defined by its functionality, message space, and encoding—as the central object, so that scheme switching becomes switching between ciphertext types. Our main FHE contribution is a batched circuit bootstrapping that lifts a GLWE ciphertext to many GGSW ciphertexts in parallel via CKKS-style bootstrapping, removing the dominant cost on the return path to GGSW and making GGSW-based leveled evaluation (LUTs, comparisons, ...) practical at scale. This is paired with a modular threshold layer that realizes the distributed key generation and decryption fixed by the FHE specification, instantiated using Shamir secret sharing over Galois rings. Both protocols are proven UC-secure against a static malicious adversary corrupting $t < n/2$ parties. The key generation provides active security with abort, while decryption has guaranteed output delivery through the use of authenticated decryption shares. Targeting NIST Threshold **Categories S4 and S5**, all constructions will be specified, open-sourced, and experimentally evaluated.

Proposed crypto-systems:

- (I) Kryptallage: A unified FHE scheme and its threshold protocol layer (Categories S4 and S5)

Keywords: Fully Homomorphic Encryption (FHE); Scheme Switching; CGGI; CKKS; Threshold Cryptography; NIST Threshold Call

Document License: [CC BY 4.0 International](https://creativecommons.org/licenses/by/4.0/)

¹Version submitted to NIST-MPTC for publication

Preview writeup. This document is provided to NIST for online publication, to foster public awareness and support public discussion within the scope of the NIST First Call for Multi-Party Threshold Schemes [NIST-IR8214C]. This “preview writeup” represents a good-faith plan for a subsequent “package submission”. However, until the deadline for package submission, the team may still modify its own composition and the submission plan, including possible changes to the technical scope, and/or the used techniques or achieved results.

Team members: Ghada Almashaqbeh ^{i1,a1}, Sonia Belaïd ^{i2,a2}, Nicolas Bon ^{i3,a2}, Ilaria Chillotti ^{i4,a3}, Mariana Gama ^{i5,a4}, Antonio Guimarães ^{i6,a5}, Erin Hales ^{i7,a6}, Hyojun Kim ^{i8,a3}, Elena Kirshanova ^{i9,a7}, Damien Ligier ^{i10,a3}, Ryan Orendorff ^{i11,a8}, Emmanuela Orsini ^{i12,a4}, Matthieu Rivain ^{i13,a2}, Luisa Siniscalchi ^{i14,a9}, Ravital Solomon ^{i15,a10}, Rick Weber ^{i16,a11}

Open Researcher and Contributor Identifiers (ORCID):

i1 (0000-0003-1297-0760); i2 (0000-0002-9437-6425); i3 (0009-0002-3777-3018); i4 (0000-0002-0319-4707); i5 (0000-0002-2759-043X); i6 (0000-0001-5110-6639); i7 (0000-0002-9551-7912); i8 (0009-0000-6477-103X); i9 (0000-0001-8924-7605); i10 (0009-0004-1759-4834); i11 (0009-0005-4903-3393); i12 (0000-0002-1917-1833); i13 (0000-0002-9855-4161); i14 (0000-0003-1813-1132); i15 (0009-0008-1424-2488); i16 (0009-0004-4226-313X)

Affiliations:

^{a1} University of Connecticut, Storrs, CT, USA,

^{a2} CryptoExperts, Paris, France,

^{a3} DESILO Inc., Seoul, South Korea,

^{a4} Department of Computing Sciences, Bocconi University, Milano, Italy,

^{a5} IMDEA Software Institute, Madrid, Spain,

^{a6} Royal Holloway, University of London, London, United Kingdom,

^{a7} Optic Solutions Systems, Abu Dhabi, UAE,

^{a8} Niobium, Dayton, OH, USA,

^{a9} Technical University of Denmark, Kongens Lyngby, Denmark,

^{a10} Fhenix, Tel Aviv, Israel,

^{a11} Databricks Inc., San Francisco, CA, USA

Main contacts:

- **Team mailing list:** contact-Kryptallage@desilo.ai
- **Primary technical contact person:** Ilaria Chillotti & Damien Ligier, Kryptallage@desilo.ai

Produced by humans. The team hereby confirms that the content in this preview writeup: (i) was produced by the team members, and (ii) was not produced by generative artificial intelligence (AI), with the possible exception of AI-proposed grammar improvements, minor integrated suggestions, or some well-identified and short localized portions of auxiliary content (e.g., some illustration); and (iii) was proofread by the team members.

1. Introduction

Kryptallage is a threshold integrated Fully Homomorphic Encryption (FHE) system in which several homomorphic encryption schemes, together with the cryptographic procedures that connect them, form a single encrypted-computation system. Encrypted values may move between schemes through explicitly specified switching procedures, so that each phase of a computation can use the representation best suited to its operations. The basic design mainly uses ciphertexts and homomorphic operations from CKKS [CKKS17] and CGGI [CGGI20], but the architecture is not restricted to the two of them.

The system is organized into two principal components.

- The *FHE computation layer* defines the supported FHE schemes, their public encryption and evaluation algorithms, and the switching procedures that connect their ciphertext and message spaces. For every supported switch, the FHE specification fixes the source and destination ciphertext types, the message transformation, the required public switching material, the admissibility conditions on the source ciphertext before switching (or decryption), and the precision, noise, and correctness guarantees of the output ciphertext. These choices determine the public and secret key properties that the threshold protocols must realize.
- The *threshold protocol layer* realizes the key-generation and decryption procedures required by the FHE computation layer. In the distributed key generation (DKG) phase, it jointly samples the secret keys according to the distributions fixed by the selected FHE schemes, generates the corresponding public and evaluation (including bootstrapping, key switching, etc.) keys, and distributes the secret material in secret shared form. The same layer implements distributed decryption (DD) and may later support recipient-specific output, share refresh, or changes of the decryption committee.

Our proposal assumes a semi-honest adversarial model for the parties performing public-key encryption and homomorphic evaluation. In contrast, for the threshold protocol layer, it assumes a malicious adversary model for the parties executing DKG or decryption phases. For encryption and evaluation phases, security against malicious parties can be achieved by attaching zero-knowledge proofs of correct encryption and/or succinct proofs of correct homomorphic evaluation, depending on the application context. We plan to investigate the feasibility of integrating such proofs into our threshold protocol layer and may include a discussion of potential solutions in the final submission.

The planned package targets **Categories S4 and S5** of the NIST Threshold Call: S4 for distributed key generation, and S5 for threshold decryption and the corresponding FHE and scheme-switching procedures [NISTIR8214C]. All submitted constructions will be specified, implemented as open source, and experimentally evaluated.

The sequence of sections below follows the structure suggested by the call: Section 2 describes the planned specification, Section 3 the open-source implementation, Section 4 the experimental evaluation, Section 5 licensing, patents, and funding, and Section 6 notes points of comparison with related submissions.

2. Specification

Organization. Kryptallage is specified as one threshold integrated FHE system consisting of an FHE computation layer and a threshold protocol layer. An optional verifiable-evaluation profile may additionally provide public verification of outsourced computation.

FHE computation layer. We introduce a scheme-switching framework that switches between *any* of the main (Ring) Learning With Errors ((R)LWE) based FHE schemes. The complete list of supported schemes will be provided in the submission. Rather than organizing this framework around named schemes (CGGI, CKKS, BGV, BFV, ...), we take the view that the relevant object is the *ciphertext type*, characterized by the functionality it supports, its message space, and its encoding, rather than by the scheme it is often attributed to. Under this lens, the major FHE schemes are recovered as particular instantiations, and switching between schemes becomes a special case of switching between ciphertext types.

Consider a ring $\mathcal{R} = \mathbb{Z}[X]/(X^N + 1)$ and the quotient rings $\mathcal{R}_q = \mathcal{R}/q\mathcal{R}$ and $\mathcal{R}_p = \mathcal{R}/p\mathcal{R}$ for some $q, p \in \mathbb{N}$ and power-of-two N , we formalize our framework around two types of ciphertexts:

- Generalized-LWE [BGV12] ciphertexts, denoted $\text{GLWE}(M)$ encrypting a message M in some plaintext space $\mathcal{R}_p$ encoded by some linear encoding function defined by the specific FHE scheme.
- Generalized-LWE-based GSW [GSW13] ciphertexts, denoted $\text{GGSW}(M)$ encrypting a message $M \in \mathcal{R}$.

Given a Generalized-GSW (GGSW) ciphertext, switching from it to almost any other ciphertext type is generally inexpensive, as GGSW provides *external products*, an operation that enables multiplications with GLWE ciphertexts while naturally preserving the encoding of the GLWE ciphertext. Building GGSW ciphertexts from other types of ciphertexts, on the other hand, is significantly more complex.

Switching from GLWE to GGSW. For a given encryption $\text{GLWE}(M)$, the process of switching to GGSW is known as *circuit bootstrapping* [CGGI20]. Existing instantiations typically rely on [CGGI20]-style bootstrapping, which bootstraps one message at a time, and leads to often-prohibitive costs of milliseconds per bit of message. We introduce a batched version of circuit bootstrapping based on an GLWE-to-GLWE bootstrapping (a new technique that we have developed specifically for this project; a manuscript will be posted online soon and construction details/proofs will be provided in the final package submission). Let $\text{ct} \in \text{GLWE}(M)$ for some $M = \sum_{i=0}^{N-1} m_i X^i \in \mathcal{R}_2$; we define the procedure $\text{liftToGGSW}(\text{ct})$, which produces a list of N GGSW encryptions $[\text{GGSW}(m_i)]_{i=0, \dots, N-1}$.

$$\text{liftToGGSW} : \text{GLWE}(M) \mapsto [\text{GGSW}(m_i)]_{i=0, \dots, N-1}$$

For efficiency, we instantiate this procedure by using ciphertexts and operations generally used for the CKKS scheme [CKKS17]. At a high level, for a ciphertext $\text{GLWE}(M)$ —but not necessarily a

CKKS ciphertext—defined over some ring $\mathcal{R}_Q$, a CKKS-style bootstrapping procedure, denoted CKKS.Bootstrap , returns $\text{GLWE}(M)$ defined over another ring $\mathcal{R}_{Q'}$ such that $Q' > Q$. Using this procedure, the main steps of our batched circuit bootstrapping are outlined below.

1. **Modulus raising and scaling at a higher Δ :** Let $\text{ct} \in \text{GLWE}(M)$; homomorphically compute a new GLWE ciphertext ct' encrypting the same message M at a large scaling factor $\Delta > Q$, over a larger modulus $Q' > \Delta > Q$. To do so we use a new bootstrapping technique based on CKKS.Bootstrap , that also reduces the noise, and is able to create space between the message rescaled at Δ and the error.
2. **Packed coefficient extraction:** by using a variant of the trace evaluation, homomorphically computes $[\text{ct}'_i]_{i=0, \dots, N/t-1}$ where each ct'_i is a GLWE ciphertext encrypting t messages m_i at the scaling factor Δ .
3. **Multiplication by the secret key:** after making a copy of the outputs of the previous step, they are homomorphically multiplied by the GLWE secret key.
4. **Windowed extraction:** Let (β, ℓ) be the decomposition parameters of the GGSW scheme, such that $\Delta > \beta^\ell$. This step homomorphically extracts the message encrypted in the ciphertexts from the two previous steps (2 and 3) at scaling factor β^j for $j = 1, \dots, \ell$.
5. **Coefficient expansion:** after a ring switching and modulus switching on all the ciphertext outputted from the previous step, this step produces N GGSW encryptions $[\text{GGSW}(m_i)]_{i=0, \dots, N-1}$ over a ring $\mathcal{R}_q$, with q a small ciphertext modulus (of the order of the ciphertexts moduli used in CGGI).

The outline above is a high-level overview without optimizations. In practice:

- The technique proposed in step 1 is a new iterative bootstrapping that is able to create more space between the message and the error, by bootstrapping the error and pushing the message higher in the most significant bits of a ciphertext with a larger ciphertext modulus.
- The windowing technique proposed in step 4 allows to extract the message previously pushed away from the noise, at a desired scaling factor. Indeed encryptions of the message encoded at different scaling factors are necessary to build the output GGSW.
- While our formal description focuses on CKKS and GGSW for generality, switching between some other schemes (e.g., BFV to CKKS) does not require generating GGSW ciphertexts, and can be implemented through more efficient variants of the protocol defined above.

Applications. Batched circuit bootstrapping provides an efficient route back to GGSW-type ciphertexts, which is one of the most costly steps in scheme switching: existing circuit bootstrapping variants produce a single ciphertext at a time, making the return path to GGSW the dominant cost whenever many selectors are needed. By producing many GGSW ciphertexts in parallel, our technique removes this bottleneck and makes GGSW-based evaluation practical at scale. This accelerates leveled evaluations relying on GGSW selectors, such as look-up tables (LUTs) and comparisons, and widens the range of computations for which switching to GGSW is worthwhile.

Threshold protocol layer. The *threshold protocol layer* realizes the key-generation and decryption protocol according to the specification fixed by the FHE computation layer.

The threshold layer is highly modular so that each protocol can be analyzed, optimized, and implemented independently. Our main ideal functionalities are the arithmetic functionality, namely $\mathcal{F}_{\text{ABB}}$, which specifies all the MPC commands used by all higher-level DKG and distributed-decryption protocols, providing arithmetic, multiplication, modulus-conversion, and opening operations on secret-shared values; the preprocessing functionality $\mathcal{F}_{\text{Prep}}$ that generates the correlated and authenticated Shamir sharing required to realize those operations.

Our concrete realization uses Shamir sharing over Galois rings, authenticated shares, reliable broadcast, and honest-majority MPC techniques. The multiplication and preprocessing phases are largely based on ATLAS-style protocols and related honest-majority constructions [GLOPS21], while the conversion procedure will use recent MPC-based modulus-conversion techniques [DKOP26]. Specific sampling procedures are selected according to the FHE-defined secret-key and noise distributions. Secret keys with independent uniform binary or ternary coefficients can be generated using standard MPC sampling techniques, whereas sparse or fixed-weight key distributions require dedicated distributed sampling protocols [AMR25].

Distributed decryption allows the output to be released publicly, delivered to a designated party, or stored in a secret shared form. Our basic protocol allows the designated output party to verify that the decryption is correct; the party verifies authenticated partial decryptions using specific verification information established during setup and reconstructs from $t + 1$ valid shares.

System Model. The system distinguishes data owners, evaluators, threshold parties and output recipients. These roles may be assigned to distinct entities.

A set of parties $\mathcal{P} = \{P_1, \dots, P_n\}$ executes the DKG. The basic implementation assumes a *synchronous network*, *authenticated point-to-point channels*, *reliable broadcast*, and a *static malicious adversary corrupting at most $t < n/2$ parties* (threshold profile (h) from [NIST-IR8214C]). In terms of the number of parties n , we will focus on the profiles (3), (S) and (M), i.e., the cases for which $3 \leq n \leq 64$. The key generation protocol provides *active security with abort*. A successful execution produces one common public output, a common setup session identifier, and a common setup transcript digest. The honest parties obtain consistent shares of the secret keys underlying that public output; if a setup check fails, the execution is discarded before the public key material is accepted.

After an accepted setup, distributed decryption provides *guaranteed output delivery* for admissible ciphertexts through the use of authenticated decryption shares. During setup, the long-lived key shares and the material used in later openings are authenticated; during decryption, malformed or missing contributions are identified and removed as erasures. Since $t < n/2$, at least $n - t \geq t + 1$ valid shares remain and these shares uniquely reconstruct the degree- t secret.

Security. The security of the employed FHE schemes resides on the hardness of the Ring Learning with Errors Problem over cyclotomic power-of-two fields. Concrete parameter choices will be tested against all relevant attacks with the help of existing tools such as the Lattice Estimator [APS15] and the Leaky LWE Estimator [DDGR20].

The security of the DKG and DD protocols is analyzed in the Universal Composability (UC) framework against a static malicious adversary corrupting $t < n/2$ parties, showing that the protocols achieve the security guarantees described above. The realization of these protocols relies

on information-theoretic Shamir privacy together with the computational assumptions required by preprocessing and authentication, including pseudorandom generators, hash functions, and any Oblivious Transfer (OT) or Vector Oblivious Linear Evaluation (VOLE) -style correlations.

3. Open-Source Implementation

1. **Code structure:** The *core code* implements the threshold-protocol layer on top of an FHE backend that provides the underlying FHE operations, as well as switch procedures. The entire codebase will be open-source under BSD-3-Clause-Clear license and will support the hardware specifications mentioned in the call.
2. **Code progress and availability:** A research prototype of the protocols exists but is **not yet publicly available**; a public, Git-compatible repository for early public testing is planned.
3. **Implementation of the networking model:** The protocol design assumes authenticated point-to-point channels together with a reliable broadcast primitive. There are several possible such primitives, and the final choice will take into account suitability for the specific use case we will study. On the baseline single-machine platform, we intend to run the parties locally and simulate these channels, possibly through virtualization. A real deployment would instantiate them with standard networking components.
4. **Testing:** The main challenges are reproducibility across platforms and faithfully exercising adversarial conditions. The use of virtualization will help to deal with these issues. We plan to test our system under adversarial behaviors and under non-optimal and pessimistic networking conditions. Concrete real-world applications showcasing the strengths of our construction will also be described in the package submission.

4. Experimental Performance Evaluation

1. **Performance:** We do not report benchmarks in this preview. We expect our switching technique from GLWE to GGSW to show good amortized performances thanks to the massive parallelization allowed by CKKS packing. Additionally, we anticipate favorable costs for the threshold protocol layer due to the efficiency of the underlying mechanisms. Quantitative comparisons with related techniques are planned for the package submission.
2. **Platform:** Our code will be compatible with the Baseline Platform mentioned in the call (a single computer with 16 cores and 64 GB of RAM, as mentioned in the call in Section 7.1 of [\[NIST-IR8214C\]](#)). A likely challenge is memory: FHE evaluation keys are large, particularly the CKKS bootstrapping key material. Another challenge is the simulation of n parties at the time, which will increase the memory footprint by a factor of n as well as quickly overwhelm the CPU capabilities. To address these challenges, we will provide proof-of-concept parameters (i.e., scaled-down parameters offering, e.g., 32-bits of security) in addition to our main parameters (offering 128-bits of security), to enable testing and verification on

the Baseline Platform. We will also benchmark our code on platforms more suited to the execution of FHE backends in the real world, with GPU capabilities and more RAM.

5. Licensing, Patent Claims, and Funding

1. The *core code* will be released under BSD-3-Clause-Clear license, and every bundled and external dependency will likewise be under an OSI/FSF-approved open-source license. We will list the exact licenses of the chosen dependencies in the package submission.
2. Following a good-faith effort to identify patents whose claims could cover the contents of the submission and that involve a team member (as inventor, applicant, or assignee) or an entity that sponsors or employs a team member, we are not aware of any such patents at this time.
3. External research funding associated with the planned submission is acknowledged as follows. This project is supported in part by Villum Young Investigator Project Number: 82457; by the Tech Europe Foundation through the SwitchFHE project; and by grants JDC2024-055789-I and CEX2024-001471-M, funded by MICIU/AEI/10.13039/501100011033 and the ESF+.

6. Comparison with Other Submissions

There are currently two other preview writeups targeting Category S5 (FHE), namely Zama and PANTHERIA. Four additional preview writeups propose Category S4 DKG protocols; since they are tailored to the requirements of their respective threshold cryptosystems rather than FHE, we restrict our comparison to the S5 submissions.

- Compared to other FHE submissions, which mainly focus on a single FHE scheme or present several schemes as independent, the strength of our contribution lies in scheme switching, combining the best of CGGI and CKKS. The construction also extends to BGV and BFV.
- The new bootstrapping we propose—which efficiently creates space between the message and the error—could also be used to perform, very efficiently, the large bootstrapping required when using full noise flooding for thresholdized decryption.
- Our threshold profile differs from Zama’s in both the corruption threshold and the security guarantees. While Zama assumes $t < n/3$ and $t < n/4$, we assume $t < n/2$. Furthermore, Zama provides guaranteed output delivery for both the DKG and distributed decryption, whereas we provide it only for distributed decryption, with the DKG offering active security with abort. PANTHERIA also supports $t < n/2$ (as well as $t < n/3$ and dishonest majority), but does not specify guaranteed output delivery for either phase of the threshold layer.
- Our DKG instantiation differs from those of the other FHE submissions and is tailored to the requirements of our FHE computation layer. Our distributed decryption instantiation relies on MPC-based modulus-conversion techniques, an approach also considered by PANTHERIA.

References

- [AMR25] Damiano Abram, Giulio Malavolta, and Lawrence Roy. “Succinct Oblivious Tensor Evaluation and Applications: Adaptively-Secure Laconic Function Evaluation and Trapdoor Hashing for All Circuits”. In: *Annual ACM Symposium on Theory of Computing*. 2025, pp. 1875–1886. DOI: [10.1145/3717823.3718164](https://doi.org/10.1145/3717823.3718164). Also at ia.cr/2025/336.
- [APS15] Martin R. Albrecht, Rachel Player, and Sam Scott. “On the concrete hardness of Learning with Errors”. In: *Journal of Mathematical Cryptology* 9.3 (2015), pp. 169–203. DOI: [10.1515/jmc-2015-0016](https://doi.org/10.1515/jmc-2015-0016). Also at ia.cr/2015/046.
- [BGV12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. “(Leveled) fully homomorphic encryption without bootstrapping”. In: *Innovations in Theoretical Computer Science Conference*. 2012, pp. 309–325. DOI: [10.1145/2090236.2090262](https://doi.org/10.1145/2090236.2090262). Also at ia.cr/2011/277.
- [CGGI20] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. “TFHE: Fast Fully Homomorphic Encryption Over the Torus”. In: *Journal of Cryptology* 33.1 (2020), pp. 34–91. DOI: [10.1007/S00145-019-09319-X](https://doi.org/10.1007/S00145-019-09319-X). Also at ia.cr/2018/421.
- [CKKS17] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yong Soo Song. “Homomorphic Encryption for Arithmetic of Approximate Numbers”. In: *International Conference on the Theory and Applications of Cryptology and Information Security*. 2017, pp. 409–437. DOI: [10.1007/978-3-319-70694-8_15](https://doi.org/10.1007/978-3-319-70694-8_15). Also at ia.cr/2016/421.
- [DDGR20] Dana Dachman-Soled, Léo Ducas, Huijing Gong, and Mélissa Rossi. “LWE with Side Information: Attacks and Concrete Security Estimation”. In: *Annual International Cryptology Conference*. 2020, pp. 329–358. DOI: [10.1007/978-3-030-56880-1_12](https://doi.org/10.1007/978-3-030-56880-1_12). Also at ia.cr/2020/292.
- [DKOP26] Ivan Damgård, Sebastian Kolby, Claudio Orlandi, and Stanislas Pawlak. “Efficient MPC-Based Modulus Conversion for Threshold FHE Decryption”. In: *IACR Cryptol. ePrint Arch*. Paper 2026/1058 (2026). URL: <https://eprint.iacr.org/2026/1058>.
- [GLOPS21] Vipul Goyal, Hanjun Li, Rafail Ostrovsky, Antigoni Polychroniadou, and Yifan Song. “ATLAS: Efficient and Scalable MPC in the Honest Majority Setting”. In: *Annual International Cryptology Conference*. 2021, pp. 244–274. DOI: [10.1007/978-3-030-84245-1_9](https://doi.org/10.1007/978-3-030-84245-1_9). Also at ia.cr/2021/833.
- [GSW13] Craig Gentry, Amit Sahai, and Brent Waters. “Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based”. In: *Annual International Cryptology Conference*. 2013, pp. 75–92. DOI: [10.1007/978-3-642-40041-4_5](https://doi.org/10.1007/978-3-642-40041-4_5). Also at ia.cr/2013/340.
- [NIST-IR8214C] Luís T. A. N. Brandão and René Peralta. *NIST First Call for Multi-Party Threshold Schemes*. (National Institute of Standards and Technology) NIST Internal Report (NISTIR) 8214C. 2026. DOI: [10.6028/NIST.IR.8214C](https://doi.org/10.6028/NIST.IR.8214C).