

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

Title: TFHE, ZHENith and Nexus

Subtitle: A Suite of Cryptosystems to Enable Fully Homomorphic Encryption Applications

Version: 0.1 (2026-01-09)¹

Team name: Zama

Team members: Mathieu Ballandras, Carl Bootland, Kelong Cong, Ben Curtis, Daniel Demmler, Tore Kasper Frederiksen, Marc Joye, Benoît Libert, Jean-Baptiste Orfila, Nigel P. Smart, Titouan Tanguy, Samuel Tap, Michael Walter

Abstract: We present three contributions: Firstly a fully specified version of the Torus Fully Homomorphic Encryption (TFHE) scheme. This is a scheme, based on variants of the Learning With Errors (LWE) assumption, which enables fast bootstrapping of ciphertexts. Secondly, we present ZHENith, a zero-knowledge proof technique, for proving the valid encryption of a TFHE ciphertext. These proofs are important in a number of areas related to using FHE in a public key setting. Thirdly, we present Nexus, a protocol suite which enables robust threshold key generation and decryption for a variety of FHE schemes (in particular BGV, BFV and TFHE). The protocols suite works in the case of honest super majority, $t < n/3$, or $t < n/4$, in the presence of malicious adversaries. We present two methods for threshold decryption, which present different tradeoffs between communication rounds and computation. To deal with all three FHE schemes, and obtain a unified framework for all such schemes, we are led to consider secret sharing, and a generic MPC functionality, over Galois Rings and not just finite fields. These components are specified within the Nexus protocol suite.

Proposed crypto-systems: TFHE: A Fully Homomorphic Encryption Scheme (S5); ZHENith: A Zero-Knowledge Proof of Correctness of TFHE Ciphertexts (S6); Nexus: A Protocol Suite for Threshold FHE (S4, S5, and S7)

Keywords: Fully Homomorphic Encryption, Zero-Knowledge Proofs, Threshold Cryptography; NIST Threshold Call



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

¹Preliminary version submitted to NIST-MPTC for review

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: Mathieu Ballandras^{i1,a1}, Carl Bootland^{i2,a1}, Kelong Cong^{i3,a1}, Ben Curtis^{i4,a1}, Daniel Demmler^{i5,a1}, Tore Kasper Frederiksen^{i6,a1}, Marc Joye^{i7,a1}, Benoît Libert^{i8,a1}, Jean-Baptiste Orfila^{i9,a1}, Nigel P. Smart^{i10,a1}, Titouan Tanguy^{i11,a1}, Samuel Tap^{i12,a1}, Michael Walter^{i13,a1}

Open Researcher and Contributor Identifiers (ORCID):

i1 (0009-0001-9603-3055); i2 (0000-0002-8390-3410); i3 (0000-0002-2636-4406); i4 (0009-0005-6377-0234); i5 (0000-0001-6334-6277); i6 (0000-0002-0358-2638); i7 (0000-0003-4433-2333); i8 (0000-0002-6914-1616); i9 (0009-0001-4526-0434); i10 (0000-0003-3567-3304); i11 (0000-0002-7965-620X); i12 (0009-0003-1778-5297); i13 (0000-0003-3186-2482)

Affiliations:

^{a1} Zama, Paris, France

Main contacts:

- **Team mailing list:** zama-nist-submission@zama.ai
- **Primary technical contact person:** Nigel P. Smart, nigel@zama.ai
- **Secondary contact person:** Titouan Tanguy, titouan.tanguy@zama.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

Our key motivation for this submission is to make available detailed specifications of the algorithms and protocols developed by Zama over the last few years. Zama is primarily a company specializing in homomorphic encryption based upon the TFHE algorithm. However, in order to deploy such TFHE-enabled applications in the real world one needs more than just an FHE algorithm.

One can view a Fully Homomorphic Encryption (FHE) application as a Multi-Party Computation (MPC) protocol. There are parties who encrypt, parties who evaluate the functions homomorphically, and parties who decrypt. Our submission will present the various components which enable such an MPC protocol to be built. In particular, the underlying FHE scheme (the TFHE scheme submitted in Category S5), the required ZKPoKs of encryption (the ZHENith proof system submitted in Category S6), and the required threshold protocols needed to secure the decryption key (the Nexus protocol suite submitted in Categories S4, S5 and S7).

Homomorphic Encryption has a number of potential applications; many of which have been explored in the academic literature. However, many of these “academic” applications are not yet ready for deployment in the real world. This is either because the performance of FHE is too slow, or the market conditions are not right, or the overall system is too complicated to be deployed in the envisioned environment.

One area where there is no such problem regarding performance, market or system, is that of blockchain. The protocols in this document have all been deployed, and are being used, in the real world in the area of blockchain. Traditionally blockchains have a state which is completely public, which causes problems as the applications requiring the use of blockchains increase. Using FHE the data held on a blockchain can be held in an encrypted form. As a blockchain’s state is long-lived, and one does not know in advance how complex, or how many, smart contracts will be applied to transform a specific encrypted value, it is imperative that any FHE scheme which is used in such a situation supports an efficient bootstrapping procedure. In addition, operations must be exact, as one does not want account balances to be “approximate”; one wants the actual value. Thus the use of an exact, bootstrappable FHE scheme such as TFHE is vital in this application domain.

One problem related to further deployment of systems based on FHE is that of standardization. Companies like to deploy cryptographic protocols that have not only passed academic peer-review, but have also been assessed by organizations such as ISO, IETF, or NIST. Currently, the TFHE scheme is going through the ISO standardization process. However, there is no standardization process for zero-knowledge proofs related to, or threshold implementations of, homomorphic encryption. Whilst the NIST threshold process does not lead to standardization, the resulting public body of reference material will helpfully give greater confidence to potential users of FHE applications.

2. Specification

Our final submission document will start with a preliminaries section which will outline FHE in general, as well as overview some basic properties of Ring-Learning-With-Errors, and other relatively standard building blocks. We will also, in the submission, give a brief overview of the popular FHE schemes of BGV [BGV12] and BFV [FV12; Bra12]. This is done in order to be able

to discuss thresholdizing these two schemes later on². Our submission will then go on to define three submitted crypto-systems: TFHE, ZHENith and Nexus. Each of these addresses a different aspect of secure computation using Fully Homomorphic Encryption. The following subsections address each of these crypto-systems, which we aim to submit, in turn.

2.1. TFHE

The first major contribution will be in the S5 category, where we will give a full specification of the TFHE [CGGI16; CGGI20; BBBB+25] FHE scheme. The TFHE scheme has very different properties from BGV and BFV. To perform any homomorphic operation one needs access to a bootstrapping functionality. On the other hand bootstrapping in TFHE is very efficient and it enables arbitrary look-up tables to be computed during the execution of the bootstrapping operation (so-called programmable bootstrapping). Indeed our main low-round threshold decryption protocol for TFHE will itself use a variant of the bootstrapping operation.

We describe a variant of TFHE which relies on a circular security assumption. The method of TFHE public key encryption that we employ is that described in [Joy24], which utilizes a Ring-LWE instance in order to generate many (standard) LWE instances. This avoids the need to have a public key which contains a large number of encryptions of zero, allowing for more compact public keys. It is also conceptually related to the public key encryption methods used in BGV and BFV.

A key aspect, which makes TFHE bootstrapping so fast, is the fact that TFHE ciphertexts are relatively small. In particular the LWE dimension and ciphertext modulus are relatively small. This means that noise distributions for the underlying LWE problems can no longer have small standard deviations, in order to ensure security via the lattice estimator [APS15].

To ensure we can generate the noise distributions efficiently within the threshold key generation of our MPC system, we avoid the use of discrete Gaussians as the noise distribution. However, even a distribution such as that used in NewHope [ADPS16] is too costly, due to the large standard deviations needed for TFHE keys and encryption. Instead we use for the noise distributions a “tweaked” uniform distribution on the interval $[-2^b, \dots, 2^b]$ for some integer value of b . This change in distribution makes very little difference to the overall security analysis of TFHE, as experiments with the lattice estimator [APS15] confirm.

2.1.1 System Model: The main algorithms of TFHE are to be implemented by three distinct entities, the encryptor, the decryptor, and the evaluator. The encryptor will apply the algorithm TFHE.Enc, the decryptor will apply the algorithms TFHE.KeyGen and TFHE.Dec, whilst the evaluator will apply the algorithms TFHE.Add, TFHE.ScalarMult, TFHE.PBS, and TFHE.SwitchSquash. The evaluator may also wish to execute the function TFHE.Enc.

For the cryptosystem TFHE all three parties are assumed to be essentially honest-but-curious. Namely, we assume the encryptor evaluates the correct encryption operation, that the evaluator evaluates the correct function on the encrypted data, and that the decryptor is fully trusted.

²Note, we are not submitting BGV or BFV for separate analysis within the scope of this call.

To overcome these limitations (on assuming semi-honest parties) one applies zero-knowledge proofs (as given in our ZHENith system), replication of the evaluator (to achieve a simple form of verifiable computation via taking an honest majority output), and applying thresholdization to the decryptor (as explained in Nexus system).

2.1.2 Security: Our specification of TFHE is very general, however in practice one usually wants a ciphertext modulus set to 2^{64} for efficiency. So we will present parameters specifically for this choice of ciphertext modulus, and two types of plaintext spaces. Our parameters result in a security level of slightly more than 128-bits of classical security, which equates to the NIST quantum computational security level of one.

2.2. ZHENith

Our second submitted cryptosystem will be in submission Category S6, and will be the ZHENith system for providing a **zero-knowledge proof** for the correct encryption of TFHE ciphertexts. Such proofs are needed to ensure that any input ciphertexts to a protocol are not adversarially generated. This is a major issue when building MPC-like protocols utilizing FHE which may have adversarially generated input, see [Sma23] for a full discussion of such protocols.

2.2.1 System Model: A zero-knowledge proof is conducted between a prover and a verifier. In our instance, as we are proving correctness of TFHE encryptions, the prover is the encryptor of the ciphertext and the verifier is anyone who wishes to verify that the encryption has been performed correctly (most likely any subsequent evaluator of the TFHE ciphertexts on specified functions). Our proofs are fully non-interactive.

2.2.2 Security: Our proofs achieve their non-interactivity by relying on hash functions (which are all modeled as Random Oracles in the security proofs) and a CRS. The CRS needs to be generated in a secure manner, thus we also provide a CRS generation ceremony that can be run between a number of parties in order to ensure that the CRS has no “backdoor” in it. This CRS ceremony is executed using a communication model similar to that in the Nexus protocols suite described below.

The zero-knowledge proofs are not post-quantum secure, as they are based on pairings. In particular the soundness property is not post-quantum secure; however the zero-knowledge property is. As such, zero-knowledge proofs are often verified as soon as they are generated, and then discarded. This provides sufficient security until a quantum computer is built. Our computational security level for soundness is chosen to be greater than 128-bits of classical security. We note that less compact, but efficient, zero-knowledge proofs could be obtained by adapting the methodology described in [LNP22; BS23; LSS24].

2.3. Nexus

Finally, in submission Categories S4, S5 and S7, we present methods for threshold decryption and key generation protocols for BGV, BFV and TFHE. The threshold protocols are built upon

a (relatively) generic, robust **MPC functionality** which works over **Galois Rings**. We have decided to place the associated MPC “gadgets” into the single Nexus submission, rather than present a separate submission in Category S7. This is due to the added complexity of the required experimental evaluation of a full-blown MPC protocol being considerable.

2.3.1 System Model: In the context of our threshold protocols, we provide fully **robust** protocols³, i.e. they can recover from injected adversarial errors, which guarantee output delivery. Our protocols utilize an **asynchronous network** in the online phase of the protocols. For some threshold profiles (for when either the number of players is large or for threshold key generation) we require an offline phase; in such situations the offline phase needs to assume a **synchronous network**. This synchronous network is emulated, over the real-life asynchronous network provided by the internet, by assuming a time-out in which any party who does not respond within a given time period is assumed to be adversarial.

For n players and t corruptions, the underlying MPC protocol in Nexus comes in two variants:

1. A variant, for when $\binom{n}{t}$ is small, which utilizes pseudo-random secret sharing. We support in this parameter region threshold decryption for BGV, BFV and TFHE. The threshold decryption works over asynchronous networks and requires $t < n/3$. For threshold key generation we require an offline phase, which also requires $t < n/3$, but which works over a synchronous network.
2. When $\binom{n}{t}$ is large, only threshold protocols for TFHE are supported; both threshold decryption and threshold key generation use an online phase which works over asynchronous networks and requires $t < n/3$. However, both also require an offline phase which requires $t < n/4$ and synchronous networks.

2.3.2 Security: As the underlying MPC protocol is generic, essentially a simple translation of the protocol of [DN07] to the Galois Ring domain, the underlying security analysis follows directly from this classical protocol. Thus the underlying MPC protocol implements the standard ideal functionality of robust-MPC. Unlike [DN07], we make, for efficiency reasons, computational assumptions. Thus our protocols, for small values of $\binom{n}{t}$ make use of so-called Pseudo-Random Secret Sharings (PRSSs). This enables highly efficient protocols to be developed utilizing relatively cheap symmetric key style cryptographic primitives such as block ciphers and hash functions. We will prove results related to the minor deviations from the blueprint of [DN07] that we make. However, for the overall security proof we refer to what is the classical literature.

3. Open-Source Implementation

3.1. Code Structure

Our core libraries implementing the crypto systems are written in **Rust**. The implementation builds upon a number of external dependencies such as **gRPC** and **tokio**. In addition, to determine

³Sometimes in the literature what we refer to as robust protocols are called protocols with Guaranteed Output Delivery (GOD-protocols)

parameters, security strengths etc, we will provide various small programs written in **Python**, **Sage** and **C++**. But these are only for the purposes of replicating our results re parameter estimation. The main programs for testing and evaluation purposes are all written in **Rust**.

3.2. Code Progress and Availability

The **Rust** code has been fully tested in production systems, and has also gone through rigorous audits by external companies. Many of the components we are submitting are already available in the open-source libraries provided by Zama. However, we will be bundling these all together with installation scripts in order to provide a single package for our submission to NIST. The submission package will be closely aligned with existing open-source documents/libraries we have made available. These include

- <https://github.com/zama-ai/tfhe-rs>
- <https://github.com/zama-ai/threshold-fhe>
- <https://eprint.iacr.org/2025/699>

3.3. Implementation of the Networking Model

In Nexus all communication between parties is managed by **gRPC** calls. We only require broadcast channels in our offline phase, which as mentioned above requires a synchronous networking model. Thus we implement, on top of the point-to-point channels provided by **gRPC** a simple broadcast protocol, based on Bracha's protocol [Bra87] (but simplified for the case of synchronous networks).

Most of the communication in Nexus can be done over authentic channels, with private channels only required in a small subset of the protocols. However, for simplicity all our point-to-point channels are implemented in a private **and** authentic manner by running **gRPC** over **TLS** connections. We needed to add a little extra logic here, as **gRPC**, when run over **TLS** channels, only ensures that messages come from an authorized as opposed to a specific party. Even when executed on a single machine in the baseline platform all communication is done via a combination of **gRPC** and **TLS**.

3.4. Testing

Reproducibility can be problematic in both TFHE and MPC implementations. We deal with both in turn and discuss our mitigation procedures in order to enable better reproducibility.

3.4.1 TFHE: The algorithms in TFHE require floating point FFT operations, which inherently have numerical errors. These numerical errors can be different on different machines due to micro-architectural differences in the underlying hardware. Whilst our error analysis ensures these errors do not affect the correctness of the overall FHE operations, they can affect the output ciphertexts. For example, a ciphertext produced by a homomorphic evaluation on one machine may be different from that on another machine, even though they both decrypt to the correct evaluation. There is little we can do to mitigate such problems without producing a crypto system which is terribly inefficient. It would appear from experimenting with various x86-based platforms,

that across x86-based machines the results are indeed the same. But when moving (say) to ARM-based platforms the outputs are indeed different.

3.4.2 MPC: Our complex MPC protocol implements each party within many interleaving threads, and thus the output of an MPC computation (whilst correct), might have intermediate values which are different on different runs (even if we fix the randomness of the parties). This is due to temporal differences in the thread scheduling on the different runs, especially if the assignment of tasks to threads is determined at run-time. This can make debugging troublesome. To mitigate against such problems we deterministically assign computational tasks to threads at compile time.

We do not provide any KAT vectors for testing the MPC sub-system as a separate system, and will only provide KAT vectors for the input/output behaviour of the MPC system on specific problem instances.

Our MPC sub-system is able to be tested under a number of malicious adversarial behaviours. We will make some of these behaviours (for example delayed or blocked messages, or incorrect messages sent) available in our testing suite.

4. Experimental Performance Evaluation

Both TFHE and ZHENith can be tested, with real-world parameters, on the baseline platform. Thus experimental results for the baseline platform will be provided in the submission.

Nexus on the other hand is harder to evaluate on the baseline platform, as the multiple parties all utilize a large number of threads and a large amount of memory. Hence, for testing purposes, we will provide a baby-parameter set with which to test Nexus on a small number of parties (e.g. four), on the baseline platform.

Experimental results will however also be presented for Nexus with more realistic numbers of parties (in the teens), in more realistic environments, with standard parameter sets. Detailed instructions as to how such experiments can be run on cloud services, such as AWS, using the Docker images which will be provided in the submission. Evaluation metrics will be provided when the AWS instances are spread across the globe. Thus allowing readers to see the performance in real-world situations.

5. Licensing, Patent Claims, and Funding

The core code will be licensed under the *BSD 3-Clause Clear License*, see <https://spdx.org/licenses/BSD-3-Clause-Clear.html>. Our final submission will also contain a list of all the publicly available Zama patent claims on the various technologies.

References

- [ADPS16] Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. “Post-quantum Key Exchange – A New Hope”. In: *USENIX Security 2016: 25th USENIX Security Symposium*. Ed. by Thorsten Holz and Stefan Savage. Austin, TX, USA: USENIX Association, August 2016, pp. 327–343. URL: <https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/alkim>. Also at ia.cr/2015/1092.
- [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.
- [BBBBB+25] Mathieu Ballandras, Mayeul de Bellabre, Loris Bergerat, Charlotte Bonte, Carl Bootland, Benjamin R. Curtis, Jad Khatib, Jakub Klemsa, Arthur Meyre, Thomas Montaignu, Jean-Baptiste Orfila, Nicolas Sarlin, Samuel Tap, and David Testé. *TFHErs: A (Practical) Handbook*. 2025. URL: <https://github.com/zama-ai/tfhe-rs-handbook/blob/main/tfhe-rs-handbook.pdf>.
- [BGV12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. “(Leveled) fully homomorphic encryption without bootstrapping”. In: *ITCS 2012: 3rd Innovations in Theoretical Computer Science*. Ed. by Shafi Goldwasser. Cambridge, MA, USA: Association for Computing Machinery, January 2012, pp. 309–325. DOI: [10.1145/2090236.2090262](https://doi.org/10.1145/2090236.2090262). Also at ia.cr/2011/277.
- [Bra12] Zvika Brakerski. “Fully Homomorphic Encryption without Modulus Switching from Classical GapSVP”. In: *Advances in Cryptology – CRYPTO 2012*. Ed. by Reihaneh Safavi-Naini and Ran Canetti. Vol. 7417. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer Berlin Heidelberg, Germany, August 2012, pp. 868–886. DOI: [10.1007/978-3-642-32009-5_50](https://doi.org/10.1007/978-3-642-32009-5_50). Also at ia.cr/2012/078.
- [Bra87] Gabriel Bracha. “Asynchronous Byzantine Agreement Protocols”. In: *Inf. Comput.* 75.2 (1987), pp. 130–143. DOI: [10.1016/0890-5401\(87\)90054-X](https://doi.org/10.1016/0890-5401(87)90054-X).
- [BS23] Ward Beullens and Gregor Seiler. “LaBRADOR: Compact Proofs for R1CS from Module-SIS”. In: *Advances in Cryptology – CRYPTO 2023, Part V*. Ed. by Helena Handschuh and Anna Lysyanskaya. Vol. 14085. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Cham, Switzerland, August 2023, pp. 518–548. DOI: [10.1007/978-3-031-38554-4_17](https://doi.org/10.1007/978-3-031-38554-4_17). Also at ia.cr/2022/1341.
- [CGGI16] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. “Faster Fully Homomorphic Encryption: Bootstrapping in Less Than 0.1 Seconds”. In: *Advances in Cryptology – ASIACRYPT 2016, Part I*. Ed. by Jung Hee Cheon and Tsuyoshi Takagi. Vol. 10031. Lecture Notes in Computer Science. Hanoi, Vietnam: Springer Berlin Heidelberg, Germany, December 2016, pp. 3–33. DOI: [10.1007/978-3-662-53887-6_1](https://doi.org/10.1007/978-3-662-53887-6_1). Also at ia.cr/2016/870.

- [CGGI20] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. “TFHE: Fast Fully Homomorphic Encryption Over the Torus”. In: *Journal of Cryptology* 33.1 (January 2020), pp. 34–91. DOI: [10.1007/s00145-019-09319-x](https://doi.org/10.1007/s00145-019-09319-x). Also at ia.cr/2016/870.
- [DN07] Ivan Damgård and Jesper Buus Nielsen. “Scalable and Unconditionally Secure Multiparty Computation”. In: *Advances in Cryptology – CRYPTO 2007*. Ed. by Alfred Menezes. Vol. 4622. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer Berlin Heidelberg, Germany, August 2007, pp. 572–590. DOI: [10.1007/978-3-540-74143-5_32](https://doi.org/10.1007/978-3-540-74143-5_32).
- [FV12] Junfeng Fan and Frederik Vercauteren. *Somewhat Practical Fully Homomorphic Encryption*. Cryptology ePrint Archive, Report 2012/144. 2012. URL: <https://eprint.iacr.org/2012/144>.
- [Joy24] Marc Joye. “TFHE Public-Key Encryption Revisited”. In: *Topics in Cryptology – CTRSA 2024*. Ed. by Elisabeth Oswald. Vol. 14643. Lecture Notes in Computer Science. San Francisco, CA, USA: Springer, Cham, Switzerland, May 2024, pp. 277–291. DOI: [10.1007/978-3-031-58868-6_11](https://doi.org/10.1007/978-3-031-58868-6_11). Also at ia.cr/2023/603.
- [LNP22] Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Maxime Plançon. “Lattice-Based Zero-Knowledge Proofs and Applications: Shorter, Simpler, and More General”. In: *Advances in Cryptology – CRYPTO 2022, Part II*. Ed. by Yevgeniy Dodis and Thomas Shrimpton. Vol. 13508. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Cham, Switzerland, August 2022, pp. 71–101. DOI: [10.1007/978-3-031-15979-4_3](https://doi.org/10.1007/978-3-031-15979-4_3). Also at ia.cr/2022/284.
- [LSS24] Vadim Lyubashevsky, Gregor Seiler, and Patrick Steuer. “The LaZer Library: Lattice-Based Zero Knowledge and Succinct Proofs for Quantum-Safe Privacy”. In: *ACM CCS 2024: 31st Conference on Computer and Communications Security*. Ed. by Bo Luo, Xiaojing Liao, Jun Xu, Engin Kirda, and David Lie. Salt Lake City, UT, USA: ACM Press, October 2024, pp. 3125–3137. DOI: [10.1145/3658644.3690330](https://doi.org/10.1145/3658644.3690330). Also at ia.cr/2024/1846.
- [Sma23] Nigel P. Smart. “Practical and Efficient FHE-Based MPC”. In: *19th IMA International Conference on Cryptography and Coding*. Ed. by Elizabeth A. Quaglia. Vol. 14421. Lecture Notes in Computer Science. London, UK: Springer, Cham, Switzerland, December 2023, pp. 263–283. DOI: [10.1007/978-3-031-47818-5_14](https://doi.org/10.1007/978-3-031-47818-5_14). Also at ia.cr/2023/981.
- [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. 2025. DOI: [10.6028/NIST.IR.8214C](https://doi.org/10.6028/NIST.IR.8214C).