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

Title: THE CLASH: THreshold ECDSA from CLASs groups linearly Homomorphic encryption

Subtitle: Threshold ECDSA from Linearly Homomorphic Encryption over Class Groups

Version: 0.1 (2026-01-12)[1]

Team name: BICYCCLIST

Team members: Cyril Bouvier, Guilhem Castagnos, Dario Catalano, Quentin Combal, Fabien Laguillaumie, Federico Savasta, Ida Tucker

Abstract: This Preview Writeup defines the starting plan for submitting THE CLASH to the NIST First Call for Multi-Party Threshold Schemes. THE CLASH is an efficient Threshold ECDSA scheme for distributing ECDSA signatures between n parties, where up to $t = n - 1$ of them can be compromised and act maliciously. THE CLASH is built upon Class Group Cryptography and it follows the paradigm of using linearly homomorphic public key encryption (PKE) to additively distribute intermediate values which sum up to the final signature. The specific PKE in this proposal is the Castagnos-Laguillaumie (CL) PKE, whose main advantage is a shorter representation than other linearly homomorphic PKEs. This characteristic is fundamental for reducing the overall communication cost. THE CLASH is accompanied by a benchmarked implementation using the BICYCL public repository, an optimized and specialized library for class group cryptography. The plan is to present the theoretical scheme with its proof of security, the building blocks and the composition of the source code, then to provide the complete package for public use in real world cases. The scheme has game based security against probabilistic polynomial time (PPT) malicious adversaries with static corruptions. The package satisfies the requirements of the NIST call, in particular interchangeability and provable security.

Categories of proposed crypto-systems: Threshold ECDSA Signature (N1.2);

Keywords: Threshold Cryptography; NIST Threshold Call; Threshold ECDSA; Class Groups

[1] Preliminary 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: Cyril Bouvier [i1,a1], Guilhem Castagnos [i2,a2], Dario Catalano [i3,a3], Quentin Combal [i4,a1], Fabien Laguillaumie [i5,a1], Federico Savasta [i6,a1], Ida Tucker [i7,a4]

Open Researcher and Contributor Identifiers (ORCID):

i1 (0009-0006-6722-4887); i2 (0009-0004-5857-7889); i3 (0000-0001-9677-944X); i4 (0009-0003-8653-9444); i5 (0000-0001-6464-1139); i6 (0009-0005-4551-1562); i7 (0000-0003-4895-5896)

Affiliations:

[a1] Université de Montpellier, CNRS, LIRMM @ Montpellier, France

[a2] Université de Bordeaux, CNRS, INRIA, IMB @ Talence, France

[a3] Dipartimento di Matematica ed Informatica, University of Catania @ Catania, Italy

[a4] PQShield, United Kingdom

Associateship clarifications:

* Ph.D. student (non-employee). † Associate (visiting researcher). ‡ Work performed while on sabbatical leave.

Main contacts:

- **Team mailing list:** bicycclist@lirmm.fr
- **Primary technical contact person:** Fabien Laguillaumie, fabien.laguillaumie@lirmm.fr
- **Secondary contact person 1:** Guilhem Castagnos, guilhem.castagnos@math.u-bordeaux.fr
- **Secondary contact person 2:** Federico Savasta, federico.savasta@lirmm.fr

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

Threshold signature schemes allow several parties to split the computation of a signature, in such a way that no single party can compute a new distributed signature by itself. In general, this can be achieved by splitting the signature key using a secret sharing protocol. The secret key is then implicitly defined but unknown to the parties joining the protocol. In threshold signature schemes, n parties jointly generate the keys, but it is sufficient that at least $t+1$ parties, for some threshold t, participate in the signing phase of the protocol to create a new signature, while $\leq t$ of them can not achieve this goal. Threshold signatures solve the "single point of failure" problem, typical in centralized signature schemes. Furthermore, a threshold signature should satisfy interchangeability with regards to signature verification. Use cases of threshold signatures include signatures on documents by a quorum, or the validation of transactions in a blockchain such as Bitcoin.

This preview document presents the main components of THE CLASH, a full threshold ECDSA signature scheme which satisfies interchangeability with regard to plain ECDSA verification. THE CLASH is a variant of the Gennaro-Goldfeder Threshold ECDSA scheme ([GG18]), and it follows the idea of building threshold ECDSA from a linearly homomorphic encryption scheme. THE CLASH is built upon the Castagnos-Laguillaumie encryption scheme (CL), whose message space can have any prime number size. This choice allows to solve the problem of slow and expensive proofs relative to the discrepancy between message space size and ECDSA curve order. This proposal is accompanied by an implementation with optimizations using the BICYCL library ([BCIL23]). This results in a very efficient scheme both in terms of computation and communication. Our threshold ECDSA proposal has provable security against active corruptions.

The package fits in Category Type N1.2: Signing, Subtype: QV, Families of Specification: ECDSA sign.

2. Specification

2.1. Organization

The specification document will include one cryptosystem, i.e., the THE CLASH threshold ECDSA protocol. With regard to the directive lines of the NIST call documentation, Sections 5,6,7, the specification document will be structured as follows.

The "Main Matter - Preliminaries" section will present the cryptographic tools necessary for understanding THE CLASH: plain ECDSA, class group cryptography, and common cryptographic primitives (such as secret sharing schemes, commitments, zero knowledge proofs). The "Main Matter - Crypto-System" section will go into details about the main building blocks used in THE CLASH.

Specifically, subsections will be dedicated to the following building blocks:

- Castagnos-Laguillaumie (CL) linearly homomorphic public key encryption scheme: we will present a short description of class group cryptography and the motivations behind the

technical choices of parameters used for the generation of the class groups of interest, the description of CL, its security, and the hard assumptions for class groups which imply that CL is secure. CL is an encryption scheme of independent interest ([CL15],[CLT18]).

- Zero Knowledge Proof of Plaintext Knowledge (ZKPoPK) for CL statements: a ZKPoPK for a CL statement allows one to prove the knowledge of a message encrypted in a CL ciphertext and the well-formedness of the latter. It does not prove knowledge of the randomness used, but suffices for our threshold ECDSA construction. These proofs, introduced and analyzed in [BDO23], [BCDLMOT24], are of independent interest. These CL based proofs are Σ-protocols made non-interactive via the well-known Fiat-Shamir transform.
- Multiplicative-to-Additive share conversion: this subprocedure allows one to convert private multiplicative shares of some value to private additive shares. It can be instantiated with any linearly homomorphic PKE, in our case the CL PKE.

2.2. System Model

Protocol overview: THE CLASH is composed of two main distributed subprotocols: a distributed key generation (DKG) and distributed signing (DSig). In THE CLASH, n parties join the DKG once and then a subset of $t+1$ of them join several DSig phases, where $t+1 \leq n$. The roles of the parties are symmetric; in DKG, parties send each other their public key for the encryption scheme and run a commit-and-reveal phase for ECDSA public key distribution. Then, they join a t-out-of-n verifiable secret sharing to compute a polynomial secret share of the ECDSA signing key. In DSig, a subset of at least $t+1$ join a 5-round signing protocol to distribute the signature. This involves a masking of the nonce which is computed by a share conversion protocol and a final reconstruction. For malicious security, signing includes a single proof for CL ciphertext well-formedness and a few discrete logarithm proofs from each party. Following Table 16 in the NIST documentation [NIST-IR8214C-2pd], the threshold profile in THE CLASH is nMfD, i.e. a medium set of n parties and $f \geq n/2$ (dishonest majority). Parameters to instantiate the PKE require the generation of a class group. THE CLASH has no trusted setup, the computation of class group parameters can be done via an interactive setup between the parties. It consists of a unique commit and reveal phase to share the randomness to be used for the generation of parameters. After that, the parties can locally compute the class group parameters using a deterministic procedure. Regarding ECDSA parameters, we use standard NIST P-curves specific to each NIST security level [FIPS:186-5].

Networking aspects: Regarding networking and communication model, we consider an unencrypted broadcast channel and secure point-to-point channels. The scheme can be built upon an unreliable channel, making it suitable for bad network conditions. The communication between parties is asynchronous.

Applicable I/O interfaces: Activation of the signature phase can be done by a simple request from any of the parties. Following the [NIST-IR8214A] documentation, THE CLASH uses implicit I/O secret-sharing modes (SSO KeyGen, Subsequent (SSI) operation). More in detail, in KeyGen,

parties run a commit-and-reveal distribution of the public ECDSA key and a Feldman Verifiable Secret Sharing (FVSS) to obtain the secret shares of the implicit distributed ECDSA signing key. In the signing phase, the parties run two Multiplicative-to-Additive (MtA) share conversion for each couple to obtain the ECDSA nonce elliptic curve point used for creating a signature. However, even if the nonce material is secret shared, this is independent of the ECDSA secret key, and the applicable I/O interfaces are (SSO) KeyGen and Subsequent (SSI) Operations.

2.3. Security

Security: THE CLASH has a game-based provable security proof against malicious adversaries with static corruptions. The protocol satisfies a definition of existential unforgeability for threshold ECDSA, which is implied by the existential unforgeability of plain ECDSA: if an adversary is able to return a forgery (r, s) on m in the threshold ECDSA protocol, the same forgery is a valid ECDSA signature on the same message m. The proof is structured in a series of hops between games, and these hops are proven using statistical arguments and the computational assumptions described in the following dedicated paragraph. Informally, due to parties' behaviour being symmetric, the scheme is not subject to trivial attacks against adaptive corruptions, but THE CLASH does not have a proof of security against adaptive corruptions. Extensions to the adaptive corruptions case exist ([CGJKR99; JL00]), but they incur an overhead. Regarding the satisfied security levels, parameters for the CL encryption scheme and for every building block/component of the protocol are consistent with the standard NIST security levels of 112, 128, 192, and 256 bits.

Adversarial Model: We assume Probabilistic Polynomial Time (PPT) adversaries. The adversaries goal is to create a forgery for the Threshold ECDSA scheme on a message m of its choice. In this cryptosystem proposal, the adversary is malicious and can perform static corruptions, i.e. it must announce which parties it corrupts at the onset of the protocol execution. It can corrupt at most t parties, where t can be up to $n-1$, i.e. it can corrupt up to all-but-one parties. We assume a rushing adversary, who can choose its round messages after seeing the messages of honest parties in the same communication round. If the threshold profile is broken, the adversary obtains more than t secret shares of the ECDSA secret key and can reconstruct the key by interpolation. This allows the adversary to create valid threshold ECDSA signatures from scratch. When $t = n-1$, this attack requires corrupting all parties.

Hard assumptions: For security to hold, we require that plain centralized ECDSA be existentially unforgeable, along with some additional assumptions. Some of these assumptions are necessary for the CL PKE, which is the main component for distributing parts of the signature in a secure way. The security of our threshold ECDSA protocol is proven under the Discrete Logarithm Problem over Elliptic Curves (DLP), Hard Subgroup Membership in the class group (HSM) [CLT18] and other hardness assumptions which are specific to class group cryptography.

3. Open-Source Implementation

Code structure: Our core code will mainly consist of the BICYCL library ([BCIL23],[BIC]), our open source C++ library for class groups based cryptography, developed and maintained by team members of the LIRMM. BICYCL implements arithmetic in the ideal class groups of imaginary quadratic fields, along with a set of cryptographic primitives based on class groups. It also implements the operations of our multi-party protocols, including the threshold ECDSA scheme we are submitting, and benchmarks for these operations. BICYCL relies on the GNU Multiple Precision Arithmetic Library (GMP) for arithmetic operations over big integers, as well as the OpenSSL library for arithmetic on elliptic curves. Both libraries are written in C. For simplicity, we plan to include GMP and OpenSSL as external dependencies, installed through the distribution's package manager. Compilation will be performed using GCC and GNU make, included as external dependencies as well. As our core code does not rely on architecture-specific features, we only select the compiler options for general optimization (e.g. `-O3`). Still, we will consider architecture-specific options if we notice significant performance improvements.

Code progress and availability: A first version of our implementation was developed in the BICYCL library in Q1 2024. Since then, fixes were added to improve performance and simplify the interface. Additional work may be needed to make the code clearer and ease the auditing process. The git repository for BICYCL is publicly available and hosted on the LIRMM's GitLab instance [BIC]. To comply with the submission requirements, we will develop additional C++ code to take care of the networking between parties. This code will rely on the BICYCL library to perform protocol operations for each party.

Implementation of the networking model: Our protocol relies on broadcast and point-to-point communication. For the latter, the protocol assumes secure channels between each pair of players, we plan to implement these channels using the Transport Layer Security (TLS) protocol. For the broadcast channel, we plan to implement it using UDP over IP. Each party will be run as a separate process, and parties will communicate through the loopback interface available on GNU/Linux systems.

Testing: We plan to have simple tests where at least one party is malicious and attempts to "cheat" at different points in the protocol. The implementation will check that in each case, the protocol is aborted by an honest party.

4. Experimental performance evaluation

Performance: The values from Table 1 are the result of running the Threshold ECDSA benchmark from BICYCL (commit `36ac9e7`). The values correspond to the mean of several experiments, 10 to 100 depending on the operation and security level, for the operations of a single party. The machine used for the measurements has the following specifications:

- x64 CPU with 16 cores (from 1.4 to 5 GHz) and 22 threads
- 16 GB of RAM
- SSD with 500 Go of memory
- Ubuntu 24.04 LTS

The resulting timings are quite stable, except for the "Setup" phase (standard deviation of $\sim$1200 ms for security parameter $\kappa = 192$) due to the generation of the class group discriminant. Network latency is not taken into account; therefore, benchmarking on multiple processes with network communication is expected to give longer timings.

Platform: The benchmark from BICYCL runs all parties consecutively in the same process and measures the time taken by a single random party. The baseline platform is an environment similar to the machine we used for Table 1, with more RAM and storage space, so it is expected to give similar timings when running the same benchmarks. But care must be taken when measuring performance with each party running concurrently on the same machine, as CPU resources have to be shared, and the machine can become overwhelmed by the intensive computation required by parties. To measure performance accurately, each party should have access to at least one CPU core when it needs it. Therefore, for our reference implementation, we plan to limit our benchmark cases to have one CPU core per party, which would give $n \leq 16$ if we don't consider hyper-threading and other system processes that run on the same machine. For cases with larger n, we will refer to the benchmarks from BICYCL.

Curve	κ	σ	n	Setup (ms)	Keygen (ms)	Signing (ms)	
				ST	ST	ST	MT
P-256	128	40	3	347	16	221	187
			5	347	17	410	325
			7	347	18	593	484
P-384	192	64	3	3 310	65	771	647
			5	3 310	81	1 432	1 145
			7	3 310	102	2 090	1 717

Table 1: Threshold ECDSA benchmark results from BICYCL, for a single party, where $t = n - 1$. The "ST" columns refer to operations performed on a single thread, and the "MT" columns to operations performed on multiple threads (up to 4 in this experiment). κ refers to the computational security parameter, and σ to the statistical security parameter.

5. Licensing, patent claims, and funding

Licenses: Our main library, BICYCL, relies on the GMP and OpenSSL libraries. Both libraries are free software and licensed under OSI approved licenses (LGPL version 3.0 for GMP, Apache license

version 2.0 for OpenSSL). BICYCL itself is licensed under GPL version 3.0 or later, a license compatible with both LGPL v3.0 and Apache v2.0. Regarding the additional code needed for the submission, we will release it under GPLv3 as well. Building uses tools from the GNU toolchain (gcc, make, ...) that are licensed under GPL. All the aforementioned licenses are approved by the Open-Source Initiative.

Patent claims: We declare that, to the best of our knowledge, no patents cover the contents of our submission. The copyright holders of our core code are the team members who contributed to its development.

Funding: The research associated with our submission is supported by the French Agence Nationale de la Recherche (ANR) project ANR-21-CE39-0006 SANGRIA, the France 2030 ANR project ANR-22-PECY-003 SecureCompute, the ANR ASTRID program under the national project AMIRAL with reference ANR-21-ASTR-0016, and by ICO, Institut Cybersecurité Occitaine, funded by Région Occitanie, France.

References

[BCDLMOT24] Lennart Braun, Guilhem Castagnos, Ivan Damgård, Fabien Laguillaumie, Kelsey Melissaris, Claudio Orlandi, and Ida Tucker. "An Improved Threshold Homomorphic Cryptosystem Based on Class Groups". In: *SCN 24, Part II*. Ed. by Clemente Galdi and Duong Hieu Phan. Vol. 14974. LNCS. Springer, Cham, September 2024, pp. 24–46. DOI: 10.1007/978-3-031-71073-5_2. Also at ia.cr/2024/717.

[BCIL23] Cyril Bouvier, Guilhem Castagnos, Laurent Imbert, and Fabien Laguillaumie. "I Want to Ride My BICYCL : BICYCL Implements CryptographY in CLass Groups". In: *Journal of Cryptology* 36.3 (July 2023), p. 17. DOI: 10.1007/s00145-023-09459-1. Also at ia.cr/2022/1466.

[BDO23] Lennart Braun, Ivan Damgård, and Claudio Orlandi. "Secure Multiparty Computation from Threshold Encryption Based on Class Groups". In: *CRYPTO 2023, Part I*. Ed. by Helena Handschuh and Anna Lysyanskaya. Vol. 14081. LNCS. Springer, Cham, August 2023, pp. 613–645. DOI: 10.1007/978-3-031-38557-5_20. Also at ia.cr/2022/1437.

[BIC] *BICYCL public git repository*. URL: https://gite.lirmm.fr/crypto/bicycl.

[CGJKR99] Ran Canetti, Rosario Gennaro, Stanisław Jarecki, Hugo Krawczyk, and Tal Rabin. "Adaptive Security for Threshold Cryptosystems". In: *Advances in Cryptology — CRYPTO' 99*. Ed. by Michael Wiener. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 98–116.

[CL15] Guilhem Castagnos and Fabien Laguillaumie. "Linearly Homomorphic Encryption from DDH". In: *CT-RSA 2015*. Ed. by Kaisa Nyberg. Vol. 9048. LNCS. Springer, Cham, April 2015, pp. 487–505. DOI: 10.1007/978-3-319-16715-2_26. Also at ia.cr/2015/047.

[CLT18] Guilhem Castagnos, Fabien Laguillaumie, and Ida Tucker. "Practical Fully Secure Unrestricted Inner Product Functional Encryption Modulo p". In: *ASIACRYPT 2018, Part II*. Ed. by Thomas Peyrin and Steven Galbraith. Vol. 11273. LNCS. Springer, Cham, December 2018, pp. 733–764. DOI: 10.1007/978-3-030-03329-3_25. Also at ia.cr/2018/791.

[FIPS:186-5] Lily Chen, National Institute of Standards, Technology, Dustin Moody, Andrew Regenscheid, and Angela Robinson. *Digital Signature Standard (DSS)*. Washington, D.C, 2023. DOI: 10.6028/NIST.FIPS.186-5.

[GG18] Rosario Gennaro and Steven Goldfeder. "Fast Multiparty Threshold ECDSA with Fast Trustless Setup". In: October 2018, pp. 1179–1194. DOI: 10.1145/3243734.3243859.

[JL00] Stanisław Jarecki and Anna Lysyanskaya. "Adaptively Secure Threshold Cryptography: Introducing Concurrency, Removing Erasures". In: *Advances in Cryptology*

— *EUROCRYPT 2000*. Ed. by Bart Preneel. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 221–242.

[NIST-IR8214A] Luís T. A. N. Brandão, Michael Davidson, and Apostol Vassilev. *NIST Roadmap Toward Criteria for Threshold Schemes for Cryptographic Primitives*. National Institute of Standards and Technology (NIST) Internal Report (NISTIR) 8214A. 2020. DOI: 10.6028/NIST.IR.8214A. URL: https://nvlpubs.nist.gov/nistpubs/ir/2020/NIST.IR.8214A.pdf.

[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.

[NIST-IR8214C-2pd] 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 2pd (Second Public Draft). March 2025. DOI: 10.6028/NIST.IR.8214C.2pd.