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

Title: Schmivitz: VOLEitH based ZK gadgets for threshold cryptography

Version: 1.0 (2026-01-20)[1]

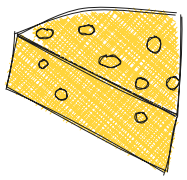
Team name: Schmivitz

Team members: Carsten Baum, Emmanuela Orsini, Peter Scholl, Benoit Razet, Marcella Hastings, Shibam Mukherjee, Christian Rechberger, James Parker

Abstract: This preview writeup represents a plan to submit Schmivitz to the NIST Threshold Call. Schmivitz is a Vector Oblivious Linear Evaluation in the Head (VOLEitH) Zero-Knowledge (ZK) system that includes two gadgets, *Weak VOLEs* and *VOLEitH*, which can be used as building blocks for cryptographic applications such as threshold signature and encryption schemes. We also define *Random All-But-One Vector Commitments* as an *internal* gadget which is a core building block of VOLEitH-proofs, but is equally important in other Zero-Knowledge Proof systems. We present an open source Rust reference implementation of Schmivitz with preliminary performance results.

Proposed crypto-systems: (I) Schmivitz: Zero-Knowledge Proof gadgets for vector commitments, weak VOLEs, and VOLEitH ZKPs (Categories S6 and S7);

Keywords: Threshold Cryptography; NIST Threshold Call



[1]Version 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: Carsten Baum [i1,a1,†], Emmanuela Orsini [i2,a3], Peter Scholl [i3,a2,†], Benoit Razet [i4,a4], Marcella Hastings [i5,a4], Shibam Mukherjee [i6,a5,a6,*], Christian Rechberger [i7,a5,a7,‡], James Parker [i8,a4]

Open Researcher and Contributor Identifiers (ORCID):

i1 (0000-0001-7905-0198); i2 (0000-0002-1917-1833); i3 (0000-0002-7937-8422); i4 (0009-0006-5698-9841); i5 (0009-0008-8217-9388); i6 (0009-0008-4399-235X); i7 (0000-0003-1280-6020); i8 (0009-0009-4399-8433)

Affiliations:

[a1] DTU Compute, Technical University of Denmark, Denmark

[a2] Department of Computer Science, Aarhus University, Denmark

[a3] Department of Computing Sciences, Bocconi University, Milano, Italy

[a4] Galois Inc., Portland, OR, USA

[a5] Graz University of Technology, Austria

[a6] Know Center, Graz, Austria

[a7] TACEO, Graz, Austria

Associateship clarifications:

[*] Ph.D. Candidate. [†] Associate Professor. [‡] Professor.

Main contacts:

- **Team mailing list:** nist-vole@galois.com
- **Primary technical contact person:** Peter Scholl, peter.scholl@cs.au.dk
- **Secondary contact person 1:** Carsten Baum, cabau@dtu.dk
- **Secondary contact person 2:** James Parker, james@galois.com

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

This submission specifies a non-interactive zero-knowledge argument of knowledge system. We define a format to specify statements, together with a corresponding witness, as well as all algorithms to parse such statements and prove knowledge of valid witnesses to verifiers. The argument system does not require a structured or common reference string, and only relies on standard, symmetric cryptography. The proof size scales linearly in the size of the statement.

1.1. Proposed Cryptosystems

We build upon the Vector Oblivious Linear Evaluation-in-the-head (VOLEitH) framework [BBSKORS23] as the foundation of the proof system. While their work focused on digital signatures with computations over Galois fields of characteristic 2 ($\mathbb{F}_{2^k}$), we allow arbitrary circuits defined simultaneously over prime fields ($\mathbb{F}_p$) and $\mathbb{F}_{2^k}$, extended with conversion gates. To specify circuits, we utilize the SIEVE Intermediate Representation (SIEVE-IR) language [SIE23].

To simplify the specification, improve modularity and due to likely interest from other submissions and cryptographic applications, we present a standalone gadget (Category S7 in [NIST-IR8214C]) and one zero-knowledge proof of knowledge (Category S6):

Weak VOLE: We formalize, specify and implement the notion of a weak Vector Oblivious Linear Evaluation correlation and its generation. Weak VOLEs are crucial to convert random vector commitments into zero-knowledge proofs in VOLEitH proof systems. As a separate gadget, they are useful to e.g. define homomorphic MACs for secure Multiparty Computation protocols.

VOLEitH: We specify and implement VOLE-in-the-Head ZK proof systems for statements over $\mathbb{F}_p$ and $\mathbb{F}_{2^k}$ and introduce modified and optimized protocols based on [BBSKORS23] and their improvements.

Towards a modular presentation, we also define an additional *internal* gadget namely **Random All-But-One Vector Commitments**: We formalize, specify and implement the notion of a random vector commitment with all-but-one opening. These are a core building block of VOLEitH-proofs (in particular, for realizing Weak VOLE), but are equally important in other Zero-Knowledge Proof systems.

The source of cryptographic hardness in all of our constructions are cryptographic hash functions as well as Pseudorandom Generators (PRGs). We will instantiate the former with SHA-3 and SHAKE, while we use AES in CTR-mode as a PRG. To argue security, we treat the hash functions as random oracles.

1.2. Real-World Pertinence

A crucial advantage of our submission over other ZK proof systems is that the security only relies on symmetric-key assumptions, that it is non-interactive and particularly efficient for statements

with "small" circuits. While there exist other ZK proof systems from similar assumptions which have an asymptotically smaller proof size and verification time (when measured in the size of the statement), ours has (i) low statement-independent cost of the proof system; and (ii) linear prover runtime with small constants.

In addition to use cases in digital signatures such as proposed in [BBSKORS23], we see a main application in privacy-preserving identification protocols. Another use case is conversion of existing threshold, cryptographic protocols to active security without introducing additional assumptions, e.g. for protocols based on lattices or code-based protocols.

2. Specification

2.1. Organization

The specification document will present the **Weak VOLE** and **VOLEitH** gadgets as building blocks for cryptographic applications such as threshold signature and encryption schemes. It will also present an *internal* gadget towards a modular presentation.

Internal Gadget: All-But-One Vector Commitment. This internal gadget is a special form of commitment scheme that allows a prover to create a commitment to a batch of random vectors, such that later, the prover can provide secret opening information to allow any verifier to learn all-but-one of the entries in each vector. The interface consists of algorithms:

- BAVC.Commit$(r, \mathsf{iv}) \rightarrow (\mathsf{com}, \mathsf{decom}, (m_1^{(i)}, \dots, m_{N_i}^{(i)})_{i=1}^{\tau})$: on input secret randomness r and an initialization vector, this outputs the commitment, opening information, and committed random vectors.
- BAVC.Open$(\mathsf{decom}, I, \mathsf{iv}) \rightarrow \mathsf{decom}_I$: on input the opening information and a set of indices $I \in [N_1] \times \cdots \times [N_\tau]$, this outputs the decommitment information for I.
- BAVC.Reconstruct$(\mathsf{com}, \mathsf{decom}_I, I, \mathsf{iv}) \rightarrow ((m_j^{(i)})_{j \in [N_i] \setminus \{I_j\}})_{i \in [\tau]})$: given a commitment, decommitment information for set $I = (I_1, \dots, I_\tau)$, and IV, this reconstructs all-but-one of the messages in the τ committed vectors.

This internal gadget is constructed via a standard approach based on GGM trees, using a length-doubling PRG and a hash function. This allows the size of a decommitment to be $O(\tau \log N)$ bits. While we do not formally define this gadget together with e.g. test vectors as we are unsure how beneficial it will be in general, we believe that presenting an *internal gadget* will modularize the presentation while potentially being useful as guidance in future use-cases.

Gadget 1: Weak VOLEs. Given a field $\mathbb{F}$, a VOLE correlation of length ℓ is a pair of random variables $(\vec{w}, \vec{u}) \in \mathbb{F}^\ell \times \mathbb{F}^\ell$ and $(\vec{v}, \Delta) \in \mathbb{F}^\ell \times \mathbb{F}$, subject to the constraint

$$w_i = v_i + u_i \cdot \Delta$$

The correlation exists between two parties, the sender $\mathcal{P}_{\text{send}}$ and receiver $\mathcal{P}_{\text{recv}}$. $\mathcal{P}_{\text{send}}$ is given $(\vec{w}, \vec{u})$, while $\mathcal{P}_{\text{recv}}$ learns $(\vec{v}, \Delta)$. The parties usually obtain their shares simultaneously. VOLE can be used to build a simple, information-theoretic MAC scheme with homomorphic properties for use in MPC protocols, such as described in [BDOZ11], or to instantiate interactive linear-size ZK proof systems such as [BMRS21; YSWW21] based on commitments.

Public verifiability for VOLE correlations, which is necessary to construct non-interactive ZK proofs, was recently addressed in [BBSKORS23]. They achieve this by observing that a weaker form of VOLE is sufficient for VOLE-based commitments. The sender obtains u_i, w_i already during the commitment phase, and can conduct arbitrary (at this point unverifiable) openings of commitments. Once all operations on commitments and their openings are completed, Δ is generated by hashing the transcript. Finally, based on this choice of Δ, the receiver learns v_i which allows it to verify the opened commitments. This means that generation of the correlation shares for $\mathcal{P}_{\text{send}}, \mathcal{P}_{\text{recv}}$ is not simultaneous, and $\mathcal{P}_{\text{send}}$ actually learns all secrets of $\mathcal{P}_{\text{recv}}$.

Weak VOLEs have not been as thoroughly researched as regular VOLEs, but they appear to be useful in other applications such as MPC protocols as well. We believe that specifying them will lead to more interest into their optimization from the research community.

We define our Weak VOLE gadget over an extension field $\mathbb{F}_{p^r}$, where the prover's $\vec{u}$ vector is defined over the base field $\mathbb{F}_p$. The interface is as follows:

- VOLE.Commit$(r, \text{iv}, \ell) \to (\text{com}, \text{decom}, \vec{u}, \vec{v})$: on input secret randomness r, an initialization vector, and a length $\ell \in \mathbb{N}$, this outputs the commitment, opening information, and the prover's random VOLE vectors $\vec{u} \in \mathbb{F}_p^\ell$ and $\vec{v} \in \mathbb{F}_{p^r}^\ell$.
- VOLE.Open$(\text{decom}, \Delta, \text{iv}) \to \text{decom}_\Delta$: on input the opening information, a challenge $\Delta \in \mathbb{F}_{p^r}$, and the IV, this outputs the decommitment information for Δ.
- VOLE.Reconstruct$(\text{com}, \text{decom}_\Delta, \Delta, \text{iv}) \to \vec{q}$: given a commitment, decommitment information for Δ, and IV, this reconstructs the verifier's VOLE output $\vec{q} \in \mathbb{F}_{p^r}^\ell$.

Gadget 2: VOLEitH. The VOLEitH gadget is a modular zero-knowledge proof system that builds on the weak VOLE gadget, together with our vector commitment internal gadget, to enable efficient and versatile ZK proofs for a broad class of circuits and constraints.

In VOLEitH, the prover ($\mathcal{P}_{\text{send}}$ in the VOLE correlation) generates the proof by first committing to the masked witness and to the randomness that will later expand into VOLE correlations, using an all-but-one vector commitment. The VOLE authentication tags are linearly homomorphic, so both the prover and the verifier ($\mathcal{P}_{\text{recv}}$ in the VOLE correlation), can perform the circuit's linear operations directly on committed data. For non-linear constraints, correctness and consistency are enforced using an information-theoretical batch check step inspired by the Quicksilver protocol [YSWW21]. Crucially, all prover messages, including QuickSilver messages, are fixed and sent before the verifier's VOLE challenge Δ is determined, yielding a public-coin transcript suitable for the Fiat–Shamir transform.

The VOLEitH framework supports multiple types of circuits. It naturally handles Boolean circuits and those over prime fields; it also supports circuits over binary extension fields, as well as efficient

conversion gates for moving between binary and larger field representations, thanks to specialized conversion gadgets based on mixed-field VOLE correlations and low-degree PRGs [ABBS25]. Finally, it extends to verifying high-degree circuit constraints, using polynomial commitment representations and optimized batch checking for such gates.

2.2. System model

The Schmivitz gadgets do not require any form of trusted setup. Since all gadgets are non-interactive, there are no networking assumptions. The prover can non-interactively produce a proof, that can be later verified by anybody who holds the proof and the same statement that was used to create the proof. The only implicit assumption of this model is that the prover and verifier have a means of agreeing upon the statement being proven, and this statement is constructed in such a way that it leaks no information about the witness beyond what is desired.

2.3. Security

Our gadgets ensure security against a malicious prover who attempts to forge proofs of false statements, as well as a malicious verifier who attempts to learn additional information about the witness. Formally, we prove that our final proof system is a zero-knowledge argument of knowledge, when modeling SHA-3 as a random oracle and AES as a secure PRG. For simplicity, we currently only target the 128-bit security level.

3. Open-Source Implementation

Code structure The reference implementation of Schmivitz is implemented as part of the Swanky [Gal19] secure computation library. The code is implemented as a Rust library and is built with standard Rust compilation tooling including the `rustc` compiler and the `cargo` package manager. The codebase is typically compiled with `cargo build --release`, tests are run with `cargo test`, and benchmarks are run with `cargo bench`. `diet-mac-and-cheese`, `mac-n-cheese-sieve-parser`, `swanky-field`, `swanky-field-binary`, and `swanky-serialization` are bundle dependencies that are other Rust libraries implemented as part of Swanky. `aes`, `sha3`, and `merlin` are Rust libraries (or crates) that serve as external dependencies that provide cryptographic primitives and Fiat-Shamir functionality. Full dependencies and compilation options are specified in the project's `Cargo.toml` Rust configuration file.

There are two main modules in the implementation. The `proof` module contains frontend circuit integration code, while the `vole` module implements the cryptographic Zero-Knowledge Proof backend. The reference implementation of Schmivitz accepts input circuits, witness values, and instance values in the SIEVE-IR format [SIE23]. By accepting this standardized format for frontend statements, the reference implementation enables users to easily encode and run their arithmetizations with VOLEitH. SIEVE-IR supports many features like multiple fields, function

```
version 2.0.0;
circuit;
@type field 2;
@begin
  $0 <- @private(0);
  $1 <- @mul(0: $0, $0);
  $2 <- @add(0: $1, $1);
  @assert_zero($2);
@end
```

Figure 1: Example SIEVE-IR circuit.

gates, conversion gates between fields, and plugins for additional functionality like RAM operations and permutation checks. Figure 1 presents a simple example SIEVE-IR boolean circuit that takes a private input and performs a multiplication, addition, and assertion. In the future, we plan to also support arithmetic circuits over large fields, and provide a Rust API so that users can programmatically define circuits in source code. This API will mirror the functionality provided by SIEVE-IR.

Code progress and availability The source code is under active development. It will be made publicly available in the Swanky Git repository[2] once we receive approval to release the code.

Implementation of the networking model: Schmivitz provides a non-interactive Zero-Knowledge proof gadget so it does not require a specific networking model.

Testing: We plan to include suitable test vectors of the proposed gadgets. For example, we will provide example SIEVE-IR circuits, witness values, and instance values as inputs for VOLEitH, as well as seeds for pseudo random number generation and output proofs. We will also include negative test cases that are expected to fail to due incorrect witness, instance values, or proofs.

4. Experimental Performance Evaluation

We report preliminary performance results by running our reference implementation on two standard circuits: AES-256 and SHA-256. The circuits are publicly available[3] in the Bristol Fashion format, which we converted to SIEVE-IR. Note that these off-the-shelf boolean circuits are not optimal and performance could be improved with more efficient circuits.

Table 1 reports the number of AND and ADD gates in the circuits, the number of VOLEs required, the prover time and verifier time in milliseconds, and the proof size in bytes for the AES-256 and SHA-256 circuits.

[2] https://github.com/GaloisInc/swanky
[3] https://nigelsmart.github.io/MPC-Circuits

Table 1: Benchmarks on AES-256 and SHA-256.

	#AND	#ADD	#VOLEs	Prover time	Verifier time	Proof size (B)
AES-256	8832	39008	9216	10.9ms	8.7ms	21680
SHA-256	22573	110644	23341	17.7ms	15.7ms	49920

The platform used to measure the performance is a Macbook Pro laptop equipped with an Apple M2 Max chip containing 12 cores and 32GB of RAM. The implementation is leveraging the AES-NI instruction set to accelerate the AES based PRG. Overall the implementation is architectured to be running single-threaded on a single core, though specific parts of the algorithm are trivially multi-threaded to increase performance. Disabling multithreading increases the running time by a factor 4.

5. Licensing, Patent Claims, and Funding

1. The core reference implementation is available under the open-source MIT license. Dependencies are available under their respective open-source licenses which include MIT, Apache 2.0, BSD2, BSD3, and Zlib.
2. There are not any known patents that cover the contents of this submission.
3. This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR001120C0085. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Defense Advanced Research Projects Agency (DARPA). Distribution Statement "A" (Approved for Public Release, Distribution Unlimited). It is also supported by the research grant VIL53029 from VILLUM FONDEN.

References

[ABBS25] Amit Agarwal, Carsten Baum, Lennart Braun, and Peter Scholl. "Low-Bandwidth Mixed Arithmetic in VOLE-Based ZK from Low-Degree PRGs". In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2025, pp. 396–426. DOI: 10.1007/978-3-031-91134-7_14.

[BBSKORS23] Carsten Baum, Lennart Braun, Cyprien Delpech de Saint Guilhem, Michael Klooß, Emmanuela Orsini, Lawrence Roy, and Peter Scholl. "Publicly verifiable zero-knowledge and post-quantum signatures from vole-in-the-head". In: *Annual international cryptology conference*. Springer. 2023, pp. 581–615. DOI: 10.1007/978-3-031-38554-4_19. Also at ia.cr/2023/996.

[BDOZ11] Rikke Bendlin, Ivan Damgård, Claudio Orlandi, and Sarah Zakarias. "Semi-homomorphic encryption and multiparty computation". In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2011, pp. 169–188. DOI: 10.1007/978-3-642-20465-4_11. Also at ia.cr/2010/514.

[BMRS21] Carsten Baum, Alex J Malozemoff, Marc B Rosen, and Peter Scholl. "Mac' n' Cheese: zero-knowledge proofs for boolean and arithmetic circuits with nested disjunctions". In: *Annual International Cryptology Conference*. Springer. 2021, pp. 92–122. DOI: 10.1007/978-3-030-84259-8_4. Also at ia.cr/2020/1410.

[Gal19] Galois, Inc. *swanky: A suite of rust libraries for secure computation*. https://github.com/GaloisInc/swanky. 2019.

[SIE23] SIEVE. *SIEVE Intermediate Representation (IR)*. https://github.com/sieve-zk/ir. 2023.

[YSWW21] Kang Yang, Pratik Sarkar, Chenkai Weng, and Xiao Wang. "Quicksilver: Efficient and affordable zero-knowledge proofs for circuits and polynomials over any field". In: *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*. 2021, pp. 2986–3001. DOI: 10.1145/3460120.3484556. Also at ia.cr/2021/076.

[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.