IBIMA Publishing
Communications of International Proceedings
https://ibimapublishing.com/p-articles/46SP/2025/4628025/
Vol. 2025 (37), Article ID 4628025

Security Analysis of Micromix: A Noncustodial Ethereum Mixer*

Kamil KACZYŃSKI[1] and Stanisław KACHEL[2]

[1]Military University of Technology, Faculty of Cybernetics,
Institute of Mathematics and Cryptology, Warsaw, Poland

[2]Military University of Technology, Faculty of Mechatronics,
Armament and Aerospace, Warsaw, Poland

Correspondence should be addressed to: Kamil KACZYŃSKI, kamil.kaczynski@wat.edu.pl

* Presented at the 46th IBIMA International Conference, 26-27 November 2025, Ronda, Spain

Abstract

This paper analyses MicroMix, a noncustodial Ethereum mixer that unlinks deposits from withdrawals using browser-side zkSNARKs, a centralised relayer, and on-chain enforcement via Semaphore and Mixer contracts. The study formalises core acceptance conditions—value conservation, nullifier uniqueness, external-nullifier scoping, and signal binding—and evaluates risks that persist despite sound cryptography, including timing correlation in small anonymity sets, Sybil pool distortion, single-relayer censorship, ETH payout liveness under gas-stipend limits, ERC-20 heterogeneity, circuit–verifier input/order mismatches, and cross-chain replay. The work proposes concrete mitigations: randomised scheduling and probabilistic batching, multi-denomination support, decentralised relayer participation with user-paid fallbacks, guarded call patterns with reentrancy protection, SafeERC20 enforcement and token whitelisting, strict public-input ordering and signal-to-field mapping, a fixed mixer-scoped external nullifier, and chain-bound proofs. With these measures, MicroMix can preserve unlinkability while improving liveness and correctness in adversarial environments, advancing practical, privacy-preserving withdrawals on Ethereum.

Keywords: Ethereum, MicroMix, blockchain, DeFi, anonymity, mixer

Introduction

As privacy-preserving transaction systems mature on Ethereum, practical mixers must balance noncustodial security, robust anonymity, and operational resilience across off-chain infrastructure and on-chain logic. MicroMix [1] addresses this by combining client-side zero-knowledge proof generation with a centralised relayer backend and a TypeScript frontend, while enforcing fixed-denomination accounting and anonymous withdrawals on-chain via Semaphore-based primitives. The solution comprises two off-chain components—a browser-based TypeScript dApp for deposits and proof construction, and a relayer backend for proof submission and fee handling—and two core smart contracts that implement identity membership, nullifier-based double-spend protection, and fixed-amount withdrawals. This paper analyses the security posture of these components, with primary focus on the Mixer.sol and Semaphore.sol contracts that govern funds flow, proof verification, nullifier management, and external nullifier policy.

This work provides a comprehensive security evaluation of MicroMix, centring on three layers: the TypeScript application (frontend) that manages EdDSA keys, witness generation, and zk-SNARK proof construction; the backend relayer service that validates, queues, and submits withdrawals while enforcing nonce safety; and the on-chain contracts (Mixer.sol and Semaphore.sol) that (i) enforce fixed-amount deposits, (ii) bind withdrawals to a recipient, relayer, and fee via a signal, and (iii) prevent double spends through nullifier recording under a fixed external nullifier tied to the mixer's address. The analysis examines correctness and safety properties across these

Cite this Article as: Kamil KACZYŃSKI and Stanisław KACHEL, Vol. 2025 (37) "Security Analysis of Micromix: A Noncustodial Ethereum Mixer " Communications of International Proceedings, Vol. 2025 (37), Article ID 4628025, https://doi.org/10.5171/2025.4628025

layers, including input validation and on-chain invariants, proof input ordering and binding, external nullifier configuration, reentrancy and ETH/ERC-20 transfer safety, trusted-setup assumptions, and censorship/DoS considerations in the relayer design. In the following sections, the architecture is dissected to identify explicit vulnerabilities, subtle specification mismatches, and operational pitfalls, followed by actionable hardening recommendations.

Micromix Architecture

MicroMix is a noncustodial Ethereum mixer that combines client-side zero-knowledge proof generation with a centralised relayer and on-chain contracts to unlink deposits from withdrawals while preserving fixed-denomination accounting and preventing double spends. Its architecture spans a TypeScript frontend for identity key management and proof construction, a backend relayer for gas abstraction and nonce-safe submission, and two core contracts—Semaphore.sol and Mixer.sol—that enforce membership proofs, nullifier uniqueness, and signal binding for recipient, relayer, and fee.

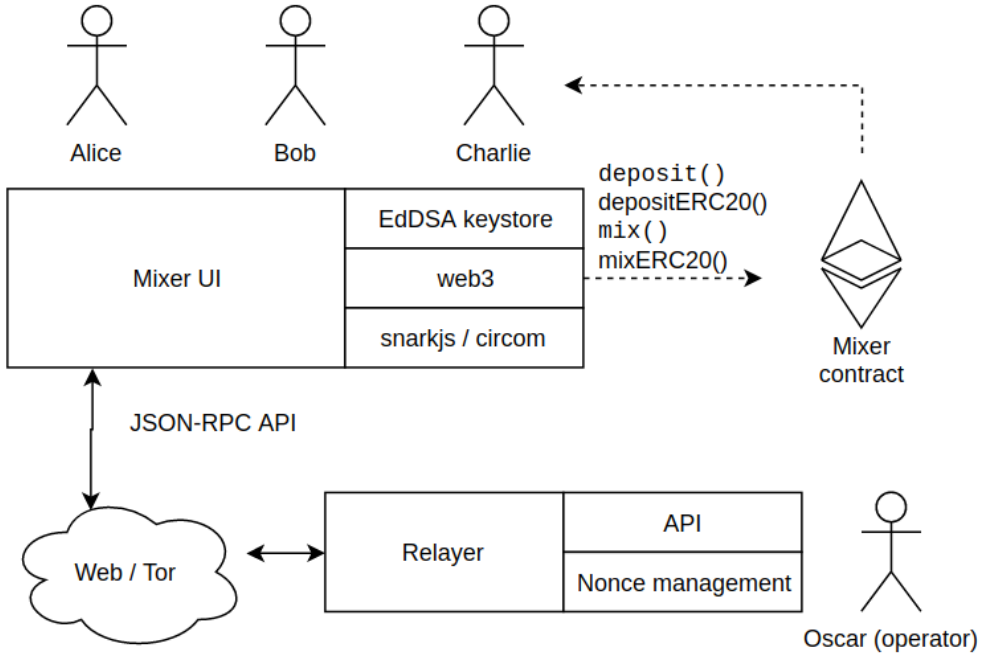

Figure 1 Overview of system components [2]

We identified the four main building blocks for the MicroMix system. These are:

- Commitment and nullifier,
- Zero-knowledge circuits,
- Merkle tree state,
- Verifier and on-chain checks.

On the implementation side, the frontend operates as the user's cryptographic workbench. It derives an identity keypair, constructs an identity commitment, and computes Merkle witnesses against the evolving set of commitments. It then generates a zero-knowledge proof attesting to membership in the set and to the uniqueness of a withdrawal via a nullifier. Crucially, it binds the intended transfer semantics into a signal that commits to the recipient address, the relayer address, and the operator fee. Formally, the application computes $signal = Keccak(recipient, relayer, fee)$, and the proof exposes its field-mapped hash as a public input while keeping the identity secret and path elements private. By deferring expensive verification to the chain and preserving confidentiality of the depositor's identity, the frontend enables a withdrawal flow where only unlinkable public artifacts appear on-chain.

The backend relayer complements the browser by abstracting gas payment and orchestrating transaction submission. It validates proofs off-chain, sequences withdrawals to manage nonce safety and liveness, and submits transactions that include the signal, proof elements, and fixed denomination constraints. While the current design

introduces a central point for censorship or delay, it also isolates users from gas funding requirements and leakage-prone networking patterns; future extensions can accommodate multi-relayer registries to diversify trust and reduce metadata concentration. Operationally, the role of the relayer is to faithfully transmit proofs that satisfy the mixer's invariants and to collect the operator fee encoded in the signal, not to custody funds.

On-chain, Semaphore.sol provides the cryptographic substrate. It maintains a Merkle tree of identity commitments and a recognised history of tree roots, exposes insertion and update routines, and records nullifier hashes to preclude withdrawal reuse. Let R denote the set of recognised roots and N the set of recorded nullifiers; acceptance of a withdrawal proof requires $root \in R$ and $nullifier \notin N$, after which the contract atomically updates N. The public inputs of the circuit follow a fixed ordering $(root, nullifierHash, signalHash, externalNullifier)$, and the verifier checks a Groth16 [3] pairing equation against a hard-coded verification key. To scope uniqueness, the external nullifier is fixed per mixer instance. In practice, Mixer.sol sets $externalNullifier = uint256(MixerAddress)$, ensuring each identity can affect at most one withdrawal under that address.

Another contract, Mixer.sol implements asset-level rules and ties proof semantics to transfers. Deposits are fixed in denomination. For ETH, the contract requires $msg.value = mixAm$. For ERC-20 tokens, it pulls exactly $mixAmt$ the tokens via $transferFrom$. Each deposit triggers insertion of the depositor's identity commitment into Semaphore, advancing the Merkle root and growing the anonymity set. Withdrawals begin with contract-side recomputation of the signal from the provided recipient, relayer, and fee. The contract then forwards the proof and public inputs to Semaphore for verification against the current or recently recognised root. If verification succeeds and the fee lies in $[0, mixAmt)$, Mixer, it pays the relayer the operator fee and transfers the remainder to the recipient. Value conservation is thus enforced at the boundary of a successful proof via the invariant $valueOut(recipient) + valueOut(relayer) = mixAmt$, and double-spend resistance is enforced via the nullifier uniqueness constraint.

Mathematically, the security of the architecture is based on four relations. First, fixed-denomination conservation holds as:

$$[\, valueOut(recipient) + valueOut(relayer) = mixAmt,]$$

with $0 \leq fee < mixAmt$ guaranteeing nonnegative payouts. Second, Merkle consistency demands that the private witness reconstructs the supplied root from a committed leaf, i.e.,

$$[\, root = MerkleRoot(identityCommitment, pathElements, pathIndex).]$$

Third, double-spend prevention is captured by the set predicate $[\, nullifierHash \notin N$ before acceptance, and $N \leftarrow N \cup \{nullifierHash\}$ after.]
Fourth, signal binding prevents fee or relayer substitution via

$$[\, signal = Keccak(recipient, relayer, fee)]$$

a circuit constraint mapping $HashToField(signal) = signalHash$. Together, these constraints ensure that a valid proof corresponds to exactly one consumable withdrawal under the mixer's external nullifier, and that the economic outcome of the transfer is uniquely determined by the signal the user committed to during proof creation.

Viewed end-to-end, MicroMix constists of a browser that crafts proofs and signals, a relayer that shoulders gas and submission risk, and a pair of contracts that rigidly enforce membership, uniqueness, and conservation. The result is an unlinkable withdrawal mechanism with fixed denominations, in which the only on-chain disclosures are a recent Merkle root, a one-time nullifier hash, and a signal-bound commitment to the transfer parameters.

Security Analysis

MicroMix was engineered to block double-spending through nullifier uniqueness and to bind withdrawals to user-chosen transfer parameters via a zero-knowledge signal. Several architectural choices and operational assumptions can still yield deanonymization or service degradation in adversarial environments. Let $mixAmt \in N_+$ denote the fixed denomination, R the set of recognised Merkle roots in Semaphore, and N the on-chain set of recorded nullifier hashes. A withdrawal proof π with public inputs $x = (root, n, s, en)$ is accepted if and only if the following predicate holds:

$$Accept(\pi, x) \Leftrightarrow (root \in R) \wedge (n \notin N) \wedge VerifyGroth16(\pi;\, x) \wedge Bind(s = Hsig(recipient, relayer, fee)) \wedge en = ENfixed$$

where $Hsig$ is a field-compatible mapping of $signal = Keccak(recipient, relayer, fee)$ and $ENfixed$ denotes the mixer's external nullifier scope, ideally $ENfixed = uint256(addr(Mixer))$. Upon acceptance, the state transition is $N \leftarrow N \cup \{n\}$ and $valueOut(relayer) = fee$, valueOut(recipient) = mixAmt − fee with $0 \leq$ fee < mixAmt, preserving $valueOut(recipient) + valueOut(relayer) = mixAmt$

Let us analyse anonymity and timing correlation. Let $A\Delta$ be the set of deposits observable in a time window Δ preceding a withdrawal and suppose all are of denomination $mixAmt$. An observer forms a posterior over candidate links by Bayes-updating a prior P0 with timing and behavioural features F (e.g., mempool presence, fee patterns), yielding

$$P(link = i \mid F, A\Delta) \propto P(F \mid link = i) \cdot P0(i)$$

with normalization over $i \in A\Delta$. If withdrawals are synchronised (e.g., immediately after midnight UTC) and the arrival rate λ of deposits is low, then $|A\Delta|$ is small and $P(link = i \mid F, A\Delta)$ becomes sharply peaked. Introducing randomized waiting $\tau \sim D$ (with entropy $H(\tau) > 0$) and probabilistic batching B of size K raises the expected $|A\Delta|$ and flattens the posterior. A simple model shows that if arrivals are Poisson with rate λ and the randomised waiting expands Δ to Δ', then the expected anonymity set size grows as

$$\mathbb{E}[|A\Delta'|] = \lambda \cdot \Delta'$$

and the chance of a perfect one-to-one timing match decay roughly as $e^{-\lambda\Delta'}$. Hence, increasing Δ' via jitter, plus multi-denomination deployment, reduces correlation probability.

Another vector is sybil inflation of the anonymity set. Let S be the number of attacker-controlled deposits and H the number of honest deposits in the active set, with total $K = S + H$. If the attacker times its withdrawals to isolate an honest target by removing attacker leaves around target events, the expected honest candidate fraction becomes

$$\rho = \frac{H}{S + H}$$

and the attacker's control over the set increases as S grows. Rate limiting can be modelled as a constraint α on per-identity deposit rate, turning S into an effective $S_{eff} \leq S$ bound by per-identity throughput. Stake- or credential-weighted admission further scales S_{eff} by a penalty factor $\beta \in 0,1)$, such that $S_{eff} = \beta \cdot S$, decreasing attacker leverage to $H/(\beta S + H)$.

The next vector possible is relayer censorship and front-running boundaries. Signal binding prevents substitution of relayer or fee: let $signal = Keccak(recipient, relayer, fee)$ and $s = HashToField(signal)$ with a fixed mapping $g: \{0,1\} * \rightarrow \mathbb{F}_p$. The circuit includes the constraint:

$$s = g(Keccak(recipient, relayer, fee))$$

and Mixer recomputes $g(Keccak(\cdot))$ on-chain. An adversary cannot alter the relayer or fee without invalidating s, i.e., for any $(relayer', fee') \neq (relayer, fee)$,

$$g(Keccak(recipient, relayer', fee')) \neq s$$

except with negligible collision probability under Keccak/g. However, a single relayer $R = \{r\}$ can censor by never submitting valid proofs. If independent relayers exist with availability vector $a = (a_1, \ldots, a_m)$, the probability of total censorship across them is

$$P_{censor} = \prod_{j=1}^{m} 1 - a_j$$

which decays as m increases, assuming at least one $a_j > 0$. A user-paid fallback after a timeout T makes $P_{censor}(T)$ drop to zero in the absence of chain-level censorship.

Let us now analyse the liveness of the ETH transfer and gas-stipend failures. Using transfer enforces a 2300 gas stipend and can fail against contracts with fallback cost $c_{gas} > 2300$, leading to a stuck-payment condition. The success predicate is

$$ok_{transfer}(recipient, x) \Leftrightarrow GasStipend \geq c_{gas}(recipient)$$

which is not guaranteed across EVM upgrades or malicious recipients. Replacing transfer with a guarded call:

$$(ok,) = recipient.call\{value: x\}("")$$

and protecting with a nonReentrant guard ensures liveness under the assumption that either recipient fallback is cheap ($ok = true$) or the system supports a pull-based retry (recipient later claims x via a safe method). Reentrancy safety is preserved by the state update order:

$$Update(N) \rightarrow Effects(fee, recipientAmount) \rightarrow Interaction(call)$$

paired with a reentrancy lock ℓ that enforces a critical section: Enter(ℓ) before Update, Exit(ℓ) after Interaction.

The next interesting point is ERC-20 invariants under nonstandard tokens. Let $B(t)$ be the token balance of the mixer. A deposit should satisfy $\Delta B = mixAmt$. Fee-on-transfer tokens induce $\Delta B = mixAmt - f_{dep}$, $f_{dep} > 0$, corrupting the invariant

$$B(t^+) = B(t^-) + mixAmt$$

On withdrawal, a fee f_{wd} yields $delivered = (mixAmt - fee) - f_{wd} < mixAmt - fee$, violating value conservation to the recipient. The safe policy is to restrict to tokens where $transferFrom$ returns $true$ and $transfer$ returns $true$ and $f_{dep} = f_{wd} = 0$ or to enforce measured accounting:

$$\Delta B_{dep} = B(t_{dep}^+) - B((t_{dep}^+) \neq mixAmt$$

reverting otherwise. For generic safety, use SafeERC20 and a whitelist W so $token \in W$ is required for deployment.

In terms of external nullifier scoping and double spends we observe that in MicroMix, uniqueness requires $en = ENfixed = uint256(addr(Mixer))$. If $ENfixed$ is unset or if multiple external nullifiers $E = \{en_1, \ldots, en_k\}$ remain active, then a malicious prover can withdraw once per en_i subject to membership constraints, violating single-withdraw semantics. The security condition is:

$$|E| = 1 \wedge E = \{ENfixed\}$$

and the Mixer must verify $en = ENfixed$ in public inputs. When satisfied, for a given identity secret ξ, the nullifier $n = Hn(\xi, ENfixed)$ becomes single use. If $en \neq ENfixed$, the proof should be rejected:

$$Accept(\pi, (root, n, s, en)) \Rightarrow en = ENfixed$$

Another common issue comes with proof input binding and mapping correctness. Let the public inputs of the circuit be ordered $x = (root, n, s, en)$, and let the on-chain verifier enforce the same order for $VerifyGroth16$. Any permutation $\sigma \neq id$ such that $VerifyGroth16(\pi; \sigma(x)) = true$ would imply a constraint mismatch between circuit and verifier keys. Regression tests must ensure:

$$\forall \sigma \neq id, VerifyGroth16(\pi; \sigma(x)) = false$$

Additionally, if s is computed as $s = floor(Keccak(signal) / 2^8)$ to fit the field (as in certain Semaphore variants), both circuit and on-chain recomputation must use the same mapping g. A proof generated with g_1 but

verified with g_2 yields either false negatives or, in pathological cases, false positives if the verifying key and constraints diverge. Correctness demands $g_{circuit} = g_{contract}$.

Let's examine the cross-chain replay resistance of MicroMix. Suppose identical verifier keys and circuits are deployed on chains A and B, with chain IDs $cA \neq cB$. If $en = uint256(addr(Mixer))$ only, then a proof referencing a root that coincidentally exists on both chains could be replayed. Binding the scope to chain via $ENfixed = Hctx(addr(Mixer), chainId)$, or adding $chainId$ as an explicit public input $x' = (root, n, s, en, chainId)$ prevents portability. The acceptance predicate tightens to $en = Hctx(addr(Mixer), chainId\_chain)$.

In the following, we formalised the DoS economics and batching. Let c_v be the expected on-chain verification gas cost per proof and c_{tx} the base transaction overhead. Submitting n proofs individually costs approximately $n(c_v + c_{tx})$. If a batching interface aggregates n proofs with amortized cost $c_{batch}(n) = c_{fixed} + n \cdot c_{marg}$, where $c_{marg} < c_v$ due to shared overhead (e.g., calldata packing, shared prechecks), the operator's break-even fee per proof is:

$$fee_{min}(n) \geq \frac{c_{batch(n)}}{n} \cdot gwei\_to\_wei(gasPrice) + margin$$

A rational fee policy sets $fee \geq fee_{min}(n)$. Pre-verification off chain reduces wasted c_v on invalid proofs. The API-rate-limiting bounds adversarial submission rate λ_a so that $\lambda_a \cdot verify\_cost\_offchain \leq backend\_budget$.

Another possible point of failure is trusted setup. Groth16 verification reduces to a single pairing product equation of the form

$$e(A, B) \cdot e(-C, \delta 2) \cdot e(\alpha 1 + x \cdot IC, \gamma 2) \cdot e(-\alpha 1, \beta 2) = 1_{GT}$$

with (A, B, C) the proof and $(\alpha 1, \beta 2, \gamma 2, \delta 2, IC)$ the verifying key. If the toxic waste from setup is known, an adversary can forge (A, B, C) satisfying the equation without a valid witness, breaking soundness and enabling arbitrary withdrawals. Therefore, correctness relies on the assumption that no participant learns trapdoor elements and that the deployed verifying key matches the circuit used by clients $VK_{onchain} = VK_{circuit}$ where equality is checked via a hash commitment or byte-for-byte match, and the frontend pins the proving key hash to the same ceremony output.

Another possible vulnerability lays in statistical deanonymization under metadata leakage. Even with cryptographic unlinkability, stable operational fingerprints increase linkability. Let features Φ include relayer identity r, fee f, gas price gp, and timing t. A classifier $h(\Phi)$ trained on historical flows can achieve success probability

$$p_s = \mathbb{E}[\, \mathbb{1}\{h(\Phi) = link,$$

which empirically increases with feature stability (low variance). Introducing randomness to fees ($f \sim U[fmin, fmax]$), diversifying relayers (larger $|R|$), and jittering submission times (higher $H(t)$) lowers mutual information $I(link; \Phi)$, which upper-bounds p_s under standard assumptions$I(link; \Phi) = H(link) - H(link \mid \Phi)$, $p_s \leq f(I(link; \Phi))$. Engineering for high $H(\Phi)$ thus reduces p_s.

Collectively, these equations formalize MicroMix's acceptance conditions, invariants, and attack preconditions. Enforcing a single chain-bound external nullifier, exact circuit–verifier input mapping, guarded ETH payouts, ERC-20 compatibility checks, multi-relayer or user-paid fallbacks, randomized scheduling, and a transparent trusted setup yields a system where $Accept \Rightarrow Conservation \wedge Uniqueness \wedge Binding \wedge Scope$ holds by construction, and where the adversary's pathways—censorship, replay, correlation, or economic DoS—are quantitatively constrained by parameters ($|R|$, $H(\tau)$, λ, c_{batch}, and ceremony soundness) that can be tuned for robust privacy and liveness.

Conclusions

MicroMix presents a solid privacy-preserving foundation built on zero-knowledge membership proofs, signal binding, and strict on-chain invariants, but its practical security depends on careful attention to anonymity set dynamics, relayer centralisation, proof/contract binding correctness, and asset-transfer mechanics. The analysis shows that while nullifier-based uniqueness and Groth16 verification enforce core safety properties on-chain, real-world adversaries can still exploit timing correlations, ERC-20 idiosyncrasies, single-relayer censorship, and configuration errors in external nullifiers and signal hashing to degrade privacy or impede liveness.

Foremost, fixed-denomination design simplifies accounting and reduces amount-based linkability, yet it increases susceptibility to timing correlation when the active deposit set is small, or when withdrawal schedules are predictable. Strengthening privacy requires increasing the expected anonymity set size and time entropy: randomised withdrawal delays, probabilistic batching, and multi-denomination support enlarge the candidate set for any observed withdrawal and flatten posterior link probabilities. Meanwhile, Sybil participation can distort the anonymity pool; rate limits, stake- or credential-weighted admission, and per-identity throughput caps reduce an attacker's effective influence and preserve honest users' cover.

Operationally, the single relayer represents a liveness and censorship chokepoint. Although signal binding prevents fee or relayer substitution, a unilateral operator can still delay or deny legitimate withdrawals. A permissionless relayer set, explicit timeouts with user-paid fallbacks, and diverse relay strategies materially reduce censorship risk and improve worst-case latency. At the asset layer, relying on ETH transfer with a gas stipend risks payout failures against contracts with nontrivial fallbacks; replacing it with guarded calls and reentrancy protection ensures robust delivery without sacrificing safety. For tokens, enforcing SafeERC20 semantics and whitelisting standard ERC-20 assets prevent invariant drift induced by fee-on-transfer or nonstandard return behaviour.

At the protocol-binding level, precise agreement between the circuit and on-chain verifier is critical. The mixer must strictly enforce the public input ordering and the exact mapping from the Keccak-derived signal into the field, and it must fix the external nullifier to the mixer's address for the lifetime of the instance. Deviations—such as multiple active external nullifiers or inconsistent signal-to-field mappings—can reintroduce double-spend or acceptance flaws. Finally, cross-chain replay can be a realistic hazard when identical circuits exist across networks, binding the scope to chain context (e.g., incorporating chain ID into the external nullifier) renders proofs non-portable.

In summary, MicroMix's cryptographic core is sound for preventing double spends and parameter-substitution front-running, and it establishes a compelling foundation for noncustodial, unlinkable withdrawals. To reach production-grade resilience, the system should: expand anonymity through randomized scheduling and potentially multiple denominations; decentralize or backstop relayer submission; harden ETH/token payout mechanics; strictly fix and verify external nullifier scope and signal-hash mapping; and bind proofs to chain context. With these refinements, MicroMix can preserve unlinkability while ensuring liveness and correctness in adversarial, high-variance environments.

References

- MicroMix source code repository, https://github.com/weijiekoh/mixer, accessed 21.09.2025
- MicroMix documentation, https://hackmd.io/qlKORn5MSOes1WtsEznu_g, accessed 21.09.2025
- Groth, J. (2016). On the size of pairing-based non-interactive arguments. In *Advances in Cryptology–EUROCRYPT 2016: 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II 35* (pp. 305-326). Springer Berlin Heidelberg.
- Mo, R., Song, H., Ding, W., & Wu, C. (2025, March). Code Cloning in Solidity Smart Contracts: Prevalence, Evolution, and Impact on Development. In *2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE)* (pp. 660-660). IEEE Computer Society.
- Wang, Z., Chaliasos, S., Qin, K., Zhou, L., Gao, L., Berrang, P. & Gervais, A. (2023, April). On how zero-knowledge proof blockchain mixers improve and worsen user privacy. In *Proceedings of the ACM Web Conference 2023* (pp. 2022-2032)
- Arbabi, A., Shojaeinasab, A., & Najjaran, H. (2025). Mixing Services in Bitcoin and Ethereum Ecosystems: A Review. *IET Blockchain*, *5*(1), e70021.
- Arango, C. C., Luna-Garcia, R., Cutchin, S., & Dagher, G. G. (2022, November). Towards deanonymization of mixing services in bitcoin. In *2022 IEEE 1st Global Emerging Technology Blockchain Forum: Blockchain & Beyond (iGETblockchain)* (pp. 1-6). IEEE.