# Sarma White Paper

by Jordan Stojanovski

## Executive Summary

Sarma introduces a groundbreaking solution for private transactions within the Ethereum Virtual Machine (EVM), leveraging Zero Knowledge (ZK) proofs for secure and private data movement across EVM blockchains. Our technology, which utilizes the encrypted data structure 'Sarma'—akin to Aleo's 'record'—enables private smart contract interactions without relinquishing the security and transparency of public blockchains. This innovation addresses significant privacy challenges in blockchain applications, enhancing both security and usability, making Sarma a pioneering solution in the realm of blockchain privacy and cross-chain functionality.

Importantly, Sarma can also travel cross-chain, enhancing bridge security by concealing the nature of the data being transferred, thereby significantly boosting overall blockchain interoperability and privacy.

Looking ahead, Sarma aims to implement idempotent transitions, allowing Sarmas to be managed off-chain effectively, significantly reducing on-chain storage needs and optimizing transactional efficiency. This innovative approach ensures that only the final results need to be stored, enhancing scalability and privacy across transactions.

## Business Model

Initially, Sarma will be introduced as a library, enabling software developers to integrate privacy-enhanced features into their applications. This approach facilitates broad adoption across various sectors by leveraging developers' existing expertise and infrastructure.

In the subsequent phase, the Sarma project will transition to a profit-making model through the integration of specialized code into Ethereum Virtual Machine pre-compiles. These pre-compiles, once deployed across different blockchains, will enhance the efficiency of on-chain verifications, reducing costs and speeding up transactions. The revenue from these operations will be derived from gas rebates, shared with the deploying blockchains and their validators, thereby creating a sustainable economic model that benefits all parties involved.

As Sarma's technology matures and integrates specialized pre-compiles into the Ethereum Virtual Machine, the applications built using the Sarma library will automatically become faster and less expensive to operate. This improvement occurs without any additional development effort from their creators, enhancing both the accessibility and economic efficiency of deploying privacy-focused applications on blockchain platforms.

## Market Analysis

The landscape for privacy-focused blockchain projects is robust, with significant investments reaching into the hundreds of millions allocated to similar initiatives such as Aleo, Aztec, and Mina. Unlike Mina and Aleo, which are developing independent Layer 1 blockchains, and Aztec, which is building a Layer 2 solution connected to Ethereum's Layer 1, Sarma uniquely

capitalizes on existing open-source Zero Knowledge proof systems like Noir, Circom, Halo3, Zokrates, and Mina. This approach allows Sarma to integrate directly with EVM blockchains, providing a seamless transition for the extensive pool of Solidity developers. Sarma's strategy reduces the learning curve significantly, positioning it as a readily accessible solution that could potentially expedite deployment compared to its competitors.

Sarma's strategic use of existing technologies not only simplifies the development process but may also accelerate its deployment relative to other projects like Aztec and Aleo. Its straightforward integration with established Zero Knowledge proof systems enhances its feasibility, potentially enabling Sarma to be operational ahead of these more complex initiatives.

# Technical Details

## Introduction

In the Ethereum blockchain and its Ethereum Virtual Machine (EVM) all data is public. However, many use cases require private transactions where the data passed from Smart Contract call to another is private and stored privately on-chain. Various privacy blockchains have emerged, achieving this feature, of which most notable are Aleo, Aztec and Mina (in alphabetical order). They utilize Cryptographic Zero Knowledge techniques to achieve this. Despite their remarkable achievements, they do not leverage on the existing EVM Solidity code base but rely on cross-chain or layer 2 transfers into the "private" realm.

Sarma is leveraging on the Aleo and Noir (by Aztec) to create private transaction and private address ability without leaving the EVM.

Aleo has achieved the public/private duality using an extremely simple primitive called *Record*. The record is an encrypted `struct` owned by a specific user, visible only to that user. Their Leo domain-specific programming language has functions with 2 parts:

- `transition`, which creates a ZK proof, which is checked on-chain. In it, the `record`s are visible.
- `finalize`, which after verifying the proof created by the `transition` it executes public code in which the `record`s are not visible unless parts of them are revealed as public inputs in the ZK proof created in the `transition`.

The `record` in Aleo resembles an Unspent Transaction Output (UTXO) in Bitcoin, but it can contain anything, not just tokens/assets. Any
information that needs privacy.

To re-confirm the validity of this idea, the upcoming Aztec Layer 2 blockchain is using a very similar technique.

Verifying ZK proofs can be done on-chain on the EVM. However, storing private data on-chain is implemented in Aleo, as they control the code base. But Sarma does not control the EVM code base, and the following section explains its implementation.

## Solution - the *Sarma*

## The Sarma rules

The Sarma corresponds to an Aleo `record`. To the EVM it's an encrypted blob of data, that the EVM can only move around, without knowing what is in it. Yet, it is not just a blob, because the EVM Smart Contract needs a permission to move this blob, which is governed by a verification of a ZK proof created off-chain. What is the condition, is up to the application developer.

As usual, the public execution is performed by the EVM Smart Contracts, which associate data to public addresses (accounts).

The private part is completely separate:
- Callers usually have private addresses (public keys). They exchange the public keys between each other privately, in secret from the other participants.
- All information processing is done inside of the ZK proof. In order to move the Sarmas around, the Solidity part has to get the permission by verifying the ZK proof on-chain as a `require` statement.
- The communication channel between the public and private

parts of the code is the set of public inputs in the ZK proof.

## Sarma exchange between callers

Even though the Sarma is not visible by the EVM, it can be decrypted off-chain and verified inside the ZK proof. For this we need to
- be able to hand over and encrypt/decrypt the Sarma from one user to another, where the users know only each other's public keys of their private addresses,
- be able to do the above non-interactively,
- the Sarma should remain encrypted (invisible) to all other participants.

Luckily, El Gamal encryption achieves this. Each Sarma creator
encrypts the Sarma with the public key of the recipient.
These participants do not need to communicate with each other to exchange the information which is otherwise invisible to all others.

To hand over the Sarma,
- The sender encrypts the Sarma using recipient's public key.
- The recipient, without communicating to the sender, decrypts the Sarma using his private key.

The recipient can decrypt the Sarma at any time, but cannot own it until the EVM Smart Contract hands it over to him, in terms of programmable logic.

## Cross-chain Sarma Transfers

While Aleo is a single Layer 1 blockchain, bridges need to be developed to transfer messages (thus assets) to other blockchains.

Aztec is a Layer 2 blockchain which relies on its sequencer to transfer messages (thus assets) to Layer 1 (Ethereum).

With Sarma, many cross-chain EVM solutions already exist, with proven track record of safety and audits. Namely, the Sarma can travel cross-chain using any of these bridges.

In addition, the Sarma makes the cross-chain bridges safer, as they do not know what they are transferring, making them immune to bribery.

# Implementation

---

## Application developer's perspective

The public part of the Smart Contracts can be written in Solidity. Sarma only provides the contract `SarmaManager` which manages the lifetime of the Sarmas. The SolidityDeveloper can simply subclass this contract and create Sarmas using `createSarma` or destroy/spend them using `destroySarma`. The Solidity contract should import the zk verifiers written in Noir and `require(verify(...))` on each sarma in order to enforce the conditions stated in the ZK proof written in Noir.

The private part of the Smart Contract can be written in Noir, as long as the library `isSarma` function is called to place the constraints that the public input Sarma has to comply with. In addition, besides calling `isSarma`, the developer can put constraints that shape other logic.

Knowing Solidity and Noir, and JavaScript for the front-end, the developers can already use Sarma to achieve private addresses and transactions on EVM and across EVMs and write such Web3 applications.

---

## What is inside?

The *private key* of the private address is a randomly generated point on the elliptic curve. The *public key* of the private address can be calculated from it, and sent to other participants or broadcasted.

The Sarma is created on the client side, encrypted with the intended recipient's public key. This encryption has to be checked (as constraint) in the ZK proof by calling `isSarma(...)`, to make sure it's a valid Sarma.

The EVM Smart Contract whenever touching the Sarma would have to call `verify(...)` which calls verification function of the ZK proof generated by the Noir's Nargo utility.

# Off-chain Sarmas - Idempotent Sarma Transitions

An *Idempotent* transition $T$:

$$\forall x : T(T(x, y), y) = T(x, y)$$

Stretching the above definition recursively as *Eventual Idempotency*, stating that a repetition of the transition in a sequence, does not change anything if one of the operations in the sequence is repeated anywhere in the sequence.

Why is this interesting? Using Sarma Transitions that are Idempotent, the intermediate Sarmas do not even need to be stored on the EVM - just the final result.

Example: Voting. Let's count the votes for a certain outcome by storing them in a Sparse Merkle Tree (SMT), and produce a recursive proof that the new voter was not in the tree, so he cannot vote twice. We do not need to make the constraints such that the proof fails if the voter tries to vote twice. Instead, we let them vote twice, but the resulting SMT to be the same as if the voter voted once. Yet, allow the proof to succeed by shaping the constraints properly. If the root of this SMT is recorded in a Sarma at the beginning we would not need to record the Sarmas on-chain each time - only at the end.

Proving Idempotency is hard, but enforcing is not. So, to implement this, one Noir function call would contain an check of Idempotency after the fact. If it passes, no Sarma is recorded on-chain, unless the user "insists". This would not be a constraint, but merely a "shortcut".

# Development Roadmap

**First Six Months**: Development of the initial usable version of Sarma, alongside a couple of sample applications to demonstrate functionality.

**Next Six Months**: Creation of a comprehensive framework of tools designed to simplify the development process. This includes tools for generating basic applications from templates, debugging tools, and additional sample applications. This phase prepares Sarma for broader developer adoption.

**Marketing Phase**: Begin marketing efforts by producing educational materials and participating in conferences and hackathons to engage with potential users and developers.

**Subsequent Development**: Focus on developing the off-chain Sarma subsystem.

**Further Development**: Focus on developing and integrating EVM verification pre-compiles, which will be marketed to various Layer 2 blockchains.

**Long-Term Expansion**: Expand the scope of Sarma to incorporate other languages like Mina PlonkyJS, Circom, Zokrates, Halo3, and others, broadening the technology's applicability and reach.

# Case Studies or Use Cases

The realm of Zero Knowledge (ZK) applications is broad and continually evolving. We will highlight a few common application patterns to showcase the versatility of Sarma. These use cases demonstrate the potential of Sarma to enhance privacy and security across various industries:

## Financial Privacy

Sarma can facilitate private financial transactions, ensuring that details of asset transfers remain confidential while maintaining the integrity of transaction verification. In addition, compliance features and tools can be built as part of
these systems to prevent unethical and illicit usage.

## Voting Systems

Employing Sarma for voting systems can ensure that votes are cast anonymously and securely, protecting voter privacy and preventing duplicate votes. Off-chain Sarmas can be used to achieve private voting by massive populations, such as
referendums in large countries.

## Supply Chain Management

In supply chains, Sarma can be used to maintain privacy of sensitive data while ensuring that all parties meet compliance and operational standards.

## Confidentiality in Business Processes

Sarma can help businesses leverage privacy to cooperate without disclosing sensitive data and confidential intellectual property to other businesses and/or the public.

## Incorruptible Decentralized Systems

Leveraging Sarma, decentralized systems and protocols can be built, which achieve better immunity to corruption and bribery.
One such protocol could be a Dark Optimistic Oracle, in which assertions can be disputed by anonymous parties, and resolution
voting can be performed by anonymous participants, incentivized anonymously, thus leaving the bad actors with no information
about "who to bribe" and whether there are any effects of such bribery.

# Team and Advisory Board

Initially the founder Jordan Stojanovski is involved in fundraising and development of the first usable version.

**Jordan Stojanovski** has extensive software development experience and has won major prizes at numerous global competitions
in the areas of Blockchain Development, Decentralized Finance (DeFi) and Zero Knowledge applications. Sarma was conceived and
prototyped by Jordan at one such competition (hackathon), wining numerous awards/prizes.

In the nearest future, we are seeking team members and advisors for:

- Advisory board.
- Fundraising.
- Marketing.
- Research and Development.
- Testing.
- Operations.

# Legal and Regulatory Considerations

As Sarma navigates the complex landscape of blockchain technology, understanding and adhering to global legal and regulatory frameworks is crucial. The project must allow for development of applications which comply with data privacy laws such as GDPR in the EU, which might affect how personal data is handled, even in encrypted forms. Moreover, the use of blockchain for financial transactions requires observance of anti-money laundering (AML) and know your customer (KYC) regulations. Regular consultations with legal experts in blockchain and data privacy laws will be essential to adapt to the evolving regulatory environment and ensure compliance. This proactive approach will help mitigate legal risks and foster a trustworthy platform for users and investors alike.

# Risk Analysis

**Regulatory Risk**: Sarma operates in a rapidly evolving regulatory environment. Adhering to international regulations such as GDPR and AML/KYC standards is crucial. Choosing the appropriate jurisdictions and investors who understand and navigate these complexities will be vital to mitigate legal and operational risks.

**Competition Risk**: The blockchain privacy sector is highly competitive, with several established and emerging technologies like Aleo, Aztec, and Mina. Maintaining technological superiority and innovating continuously are necessary to stay ahead.

**Adoption Risk**: The success of Sarma hinges on its adoption by developers and acceptance by users. There's a risk that the complexity of ZK proofs or resistance to new technologies might hinder widespread adoption. Strategic marketing and robust developer support are essential to overcoming these challenges.

# Appendices

## Appendix 1: Frequently Asked Questions

**Q:** The Sarmas are associated with addresses, does not this break the privacy?
**A:** Not at all. The public execution can be called by public addresses, but the private addresses do not have to be associated to them. In addition, the public Solidity code can "mint" Sarmas seemingly "out of thin air", but the private Noir code governs the validity of the program logic, as long as the Solidity code verifies the Sarma by calling `verify`.

**Q:** What does Sarma mean?
**A:** Sarma was conceived at the ETHGlobal Istanbul 2023 hackathon. In Turkish "sarma" means "wrapping". We use the term to denote an encrypted controlled record. In Macedonian (the word coming from the Ottoman Empire) "sarma" ("сарма") is a tasty dish made of rice and meat wrapped in pickled cabbage leaves.

**Q:** Is the execution of Sarma fast?
**A:** Yes, there is no on-chain proving, thus no on-chain heavy computation. Only verifications occur on-chain, and there is a helpful pre-compile already in most EVMs to make the verification fast. Executing a small amount of public code is faster, but for private transactions,

as the code is executed off-chain, the execution happens in constant time and cost, regardless of the amount of calculations.

**Q:** How come we do not need a tree to store the Sarmas, as Aleo stores its "records" in a Sparse Merkle Tree?
**A:** The Sarmas are stored and governed by the EVM Smart Contract. The EVM is responsible for storing them. However, in order to hide the Sarmas the private and the public address spaces are separated. Even though the movement of Sarmas can be observed, the EVM Smart Contract acts as a "mule" and does know what it is moving. In addition to this, each function call to the EVM Smart Contract can be performed from a different address (EOA). Actually some knowledge about the Sarma movements is revealed, but this knowledge is useless and thus irrelevant.

**Q:** Are private applications illegal or unethical?
**A:** Not if they are written for a good cause. For example, secret voting gives privacy to the voters in order to protect them
from targeting or persecution. Secret optimistic oracles achieve better immunity against resolution voter corruption.

**Q:** Is Sarma a tool for evading financial compliance?
**A:** No, Sarma is merely a technological infrastructure, just like Zero Knowledge proof systems and other cryptography tools. It is up to the application developer to make sure the product they build complies with ethics, rules and laws. For example Zero Knowledge mixers can be used for illicit fund transfers as well as legitimate privacy purposes (one would not want to reveal all account balances and transactions upon each purchase, as this would attract potential criminals and other predatory actors). However, the developer can implement a legitimate mixer with "reveal key" feature, so upon request by law enforcement, the user can clear demonstrate legitimacy by revealing the transactions only to the authorities and not to other "prying eyes". Such privacy mixer, may be even built to seek pre-approval from legal authorities prior to usage.

# Appendix 2: References


- ETHGlobal Istanbul 2023 hackathon project "Sarma" where it was conceived and prototyped: https://ethglobal.com/showcase/sarma-evm-zkevm-pexc-9wp7o

- ETHGlobal Istanbul 2023 finalist presentation where "Sarma" was presented and awarded: https://www.youtube.com/watch?t=2609&v=RW6qZTEIqWc&feature=youtu.be

- Aztec Noir documentation: https://noir-lang.org/docs/

- ZEXE articles: https://medium.com/zeroknowledge/what-is-zexe-part-i-5fd27b8dcf96, https://medium.com/zeroknowledge/what-is-zexe-part-ii-bb24b560aebd, https://medium.com/zeroknowledge/what-is-zexe-part-iii-a5515483bbeb

- ZEXE paper: https://eprint.iacr.org/2018/962.pdf