Composite 2: Cryptography

Nesa's cryptographic methods for secure and verifiable computation pair with its hardware composite to provide versatility in the execution environment based on the size of the model, the preference of the user, and the complexity of the task. The amalgamation of the two composites is such that each can in some capacity offset the other's limitations. The resource-intensive nature of cryptographic solutions leads Nesa to default to their usage, particularly for small-sized AI tasks requested through the AIVM. This minimizes the computational overhead that cryptographic privacy-preserving techniques could face on a large model.

As a result, our cryptographic solution employs *broadcast secret sharing* (*BSS*), a protocol that enables a user to distribute a public message that allows a group of recipients to obtain secret shares of a confidential message without revealing the message itself.

Nesa operates on BSS to ensure that at no point is the unencrypted input data exposed to any of the participating nodes. Instead, secret-shared inputs are used to perform SMPC, allowing each node to process the data with its model and generate an individual inference result.

This approach maintains the privacy of both the input and the model. This section describes the steps of result submission, majority vote, rewards, and punishment that comprise the cryptography composite.

Computation on Fragmented Data

By way of this hybrid composite, Nesa allows the user to encode their sensitive input data into a format suitable for broadcast secret sharing. The user's data is transformed into a public message consisting of multiple secret shares, each intended for a different computation node. When nodes receive their respective shares, they perform computations on this fragmented data without the ability to reconstruct the original input individually.

Through this process, nodes collaboratively engage in the computation of the AI inference task using an SMPC protocol, where each node contributes its computational power while the data remains distributed and confidential.

Protocol Steps

We break down Nesa's cryptography composite in the following steps:

1. Input Data Preparation: The user prepares their input data encrypted by a secret *s.* The secret *s* is processed into a public message using a broadcast secret-sharing scheme. The user submits the ciphertext *c* and the public message to the blockchain along with the inference request.
2. Node Committee Formation: Similar to the process outlined in Chapter 3, a committee of nodes is selected based on a VRF technique to ensure a random and fair choice of participants.
3. Secure Multi-Party Computation: The nodes run the BSS algorithm collectively to obtain a (t, n)-threshold secret sharing of the secret *s*. Upon receiving their secret shares of the secret, the nodes initiate an SMPC protocol. Each node processes the input with its own AI model, yielding an inference result revealed to itself. If we let [·] denote something that is secret shared, then this protocol contains several sub-steps: 1) [m] ⟵ Decrypt (*c,* [*s*]); 2) [R] ⟵ LLMi([m]); 3) R ⟵ Reveal(*i*, [*R*]).
4. Result Commitment: Nodes commit to their results by sending a cryptographic commitment to the blockchain, ensuring commitment to the individual computations.
5. Result Reveal: After a specified time frame, nodes reveal their results by submitting them to the blockchain, alongside any necessary cryptographic proofs of correctness.
6. Result Aggregation and Majority Formation: A smart contract aggregates the partial results, applying the majority vote strategy to determine the outcome of the inference task. The smart contract identifies the consensus result based on the majority of matching partial results.
7. Reward and Punishment Enforcement: The smart contract dispenses rewards to nodes that contributed to the consensus result. Nodes that submitted deviating results are penalized as outlined in the system's rules.
8. Inference Result Finalization: The consensus result, representing the secure and private computation of the AI inference task, is recorded on the blockchain. Users can then retrieve and make use of the final result for their purposes.

Beyond ensuring the confidentiality of input data through broadcast secret sharing, Nesa utilizes ZKPs to verify that nodes have correctly performed their part of the secure multi-party computation without revealing the underlying data or model parameters.

The zero-knowledge scheme on Nesa for proving AI model computation (zkAI) consists of the following algorithms:

- pp ← zkAI.KeyGen(1^λ): Given the security parameter, the algorithm generates the public parameters pp.
- com^W ←zkAI.Commit(W, pp,r): The algorithm commits the parameters W of the model using the randomness r.
- (y,π)←zkAI.Prove(W, X, pp,r): Given a data sample X, the algorithm runs an inference algorithm to get y = pred(W, X) and generates the proof π.
- {0,1}←zkAI.Verify(comW, X,y,π, pp): The algorithm verifies the prediction y with the commitment com W, the proof π, and the input X.

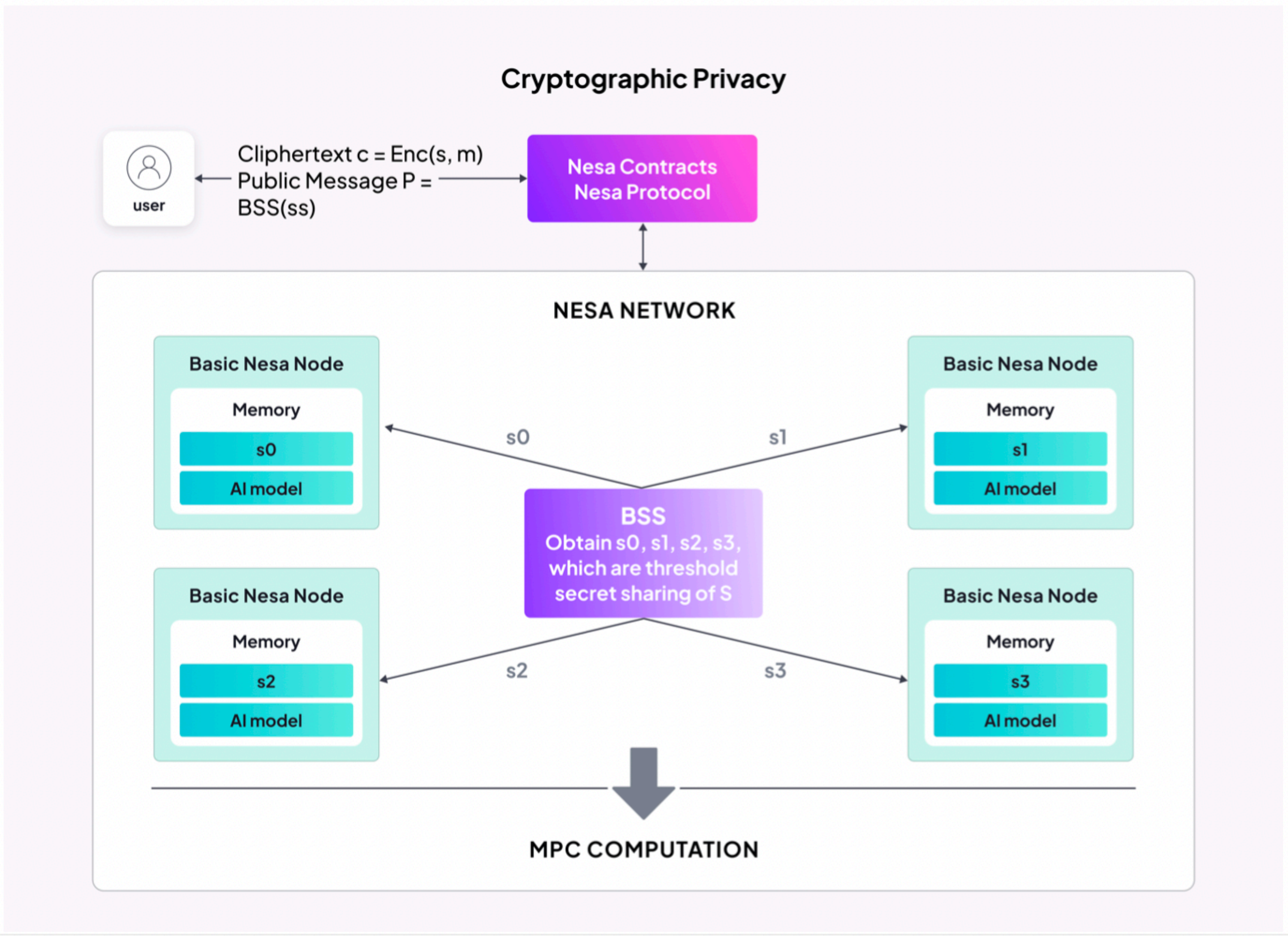
Recent efforts in the field have yielded a specialized protocol, called the zkCNN, which focuses on the verification of neural network evaluations within the zero-knowledge proof framework. This protocol enables the construction of proofs that attest to the correct execution of computations related to AI models, such as convolutional neural networks, without compromising data privacy.

The adoption of ZKP, however, for verifiable computation within AI model evaluation comes not without significant computational overhead. The generation and verification of zero-knowledge proofs for complex neural network operations are a resource-intensive process, which can lead to prolonged execution times and increased costs.

As a result, Nesa orchestrates the application of ZKP through Split-Flow when it ascertains the requested query is on a relatively small AI model, where the trade-off between the added security and the computational burden is justifiable to the user.

Our system strategically incorporates ZKP-based verifiable computation for use cases where the model complexity and inference task size allow for the practical application of these advanced cryptographic methods.

This selective integration ensures enhanced privacy and verifiable computation when appropriate without disproportionately affecting the system's overall throughput and performance, safeguarding scalability. Split-Flow is the arbiter protocol that is constantly evaluating and calibrating the security-utility trade-off for each inference request and AIVM kernel activated on Nesa.



Nesa's Cryptography Composite. A user submits ciphertext in the request transaction, along with a public message that enables threshold sharing of a secret. The ciphertext can be decrypted in the committee of nodes, through broadcast secret sharing and threshold decryption. The decryption reveals the plaintext, enabling each node to continue performing subsequent inference computation. The decryption can alternatively reveal only plaintext shares such that subsequent computation is carried in MPC, but this is more computationally heavy and is orchestrated by preset for small- scale models.

Previous
Composite 1: Hardware

Next
LLMs in the Nesa Ecosystem

Last updated 1 month ago