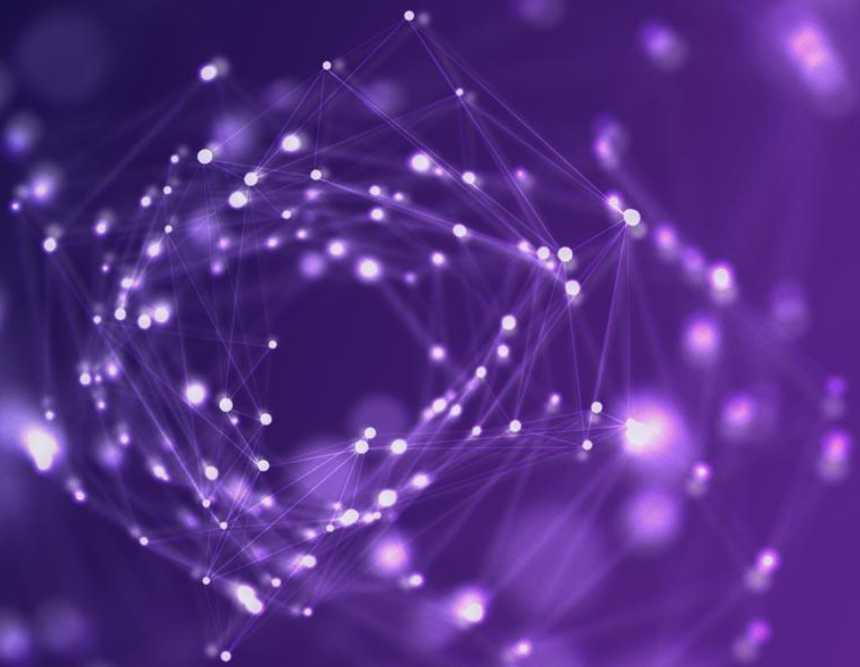
Nahmii 2.0 White Paper



Mark Briscombe

Dr. John Derbyshire

Dr. Jens Ivar Jørdre

Jacobo Toll-Messia

July 7, 2021

Contents

Introduction

Developed and launched in 2018, Nahmii 1.0 represented the first commercial implementation of Ethereum layer-2 scaling. It served as a showcase for our 'state pool' technology and the basis for a number of innovative products. Building on customer feedback from the past two years, alongside new developments in the Ethereum ecosystem, we can now announce 'Nahmii 2.0'.[1]

The Nahmii protocol was designed to solve a fundamental problem in the blockchain space: the need for a high-performance, low-cost scaling solution. It sits as a scalable layer built on top of Ethereum, leveraging the layer 1's class-leading security, whilst enabling supreme performance. Crucially, Nahmii addresses all requirements for *commercially-relevant* scaling:

- **Transaction throughput**: A key goal of all scaling solutions: enables a far greater number of transactions per second. Across the network, Nahmii has no limit to the number of transactions per second, with performance comparable to an optimised database

- **Low latency**: Reduces the time it takes for transactions to be processed and gives users immediate feedback. Latency on Nahmii is measured in milliseconds

[1] For the remainder of this document, we refer to 'Nahmii 2.0' as 'Nahmii'.

- **Instant finality**: Often overlooked, but especially key for financial applications to eliminate execution risk. Transactions on Nahmii have instant finality

- **Low fees**: Provides general viability to all use cases, including micro-transactions. Nahmii's fees will always be low

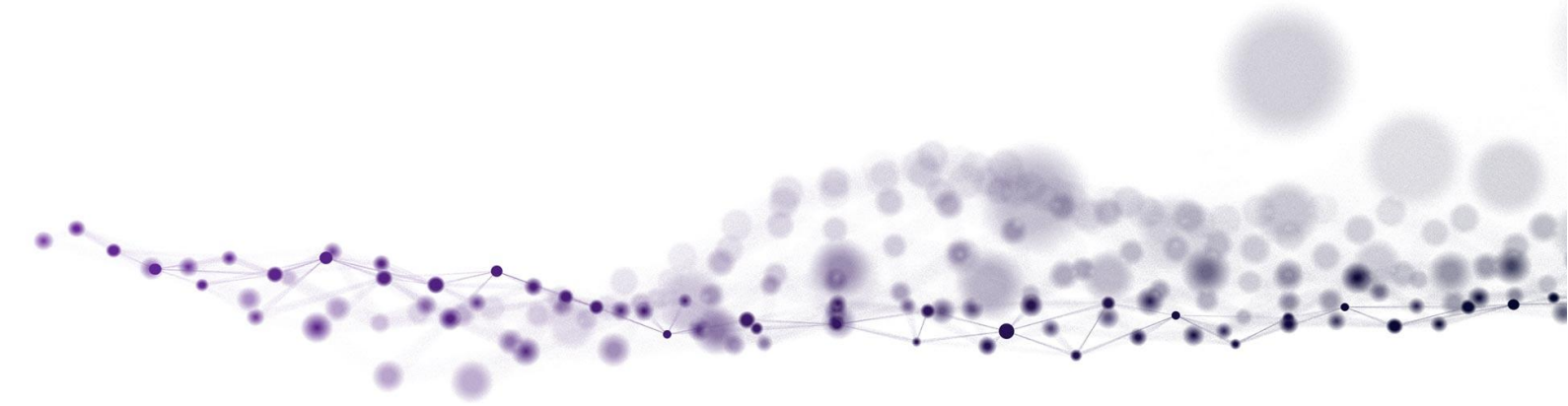
- **Predictable fees:** Essential for managing and planning the costs of a product. There will be no variability of fees on Nahmii

With Nahmii 2.0, we will retain all of these key advantages while incorporating exciting new features and optimisations based on real-world feedback.

With almost four years of experience in building and refining layer-2 scaling solutions, we are singularly equipped to apply our learnings to craft the market-leading protocol. This new version of Nahmii will utilise the same proprietary 'state-pool' architecture, whilst supporting generalised, EVM-compatible smart contracts.

The addition of generalised smart contracts in Nahmii 2.0 will bring significant benefits to the entire Ethereum ecosystem. Currently, smart contracts on the Ethereum blockchain are expensive, slow and difficult to use. With Nahmii, smart contracts will maintain their innovative security and trustless characteristics while gaining a radical performance boost. Taken together, the improvements to throughput, latency, finality and cost will bring smart contracts to the masses.

In this document, we describe the characteristics of Nahmii, its architecture and the considerations towards security. We also touch on some of the serious flaws present in other layer-2 offerings.



Our History in Scaling Ethereum

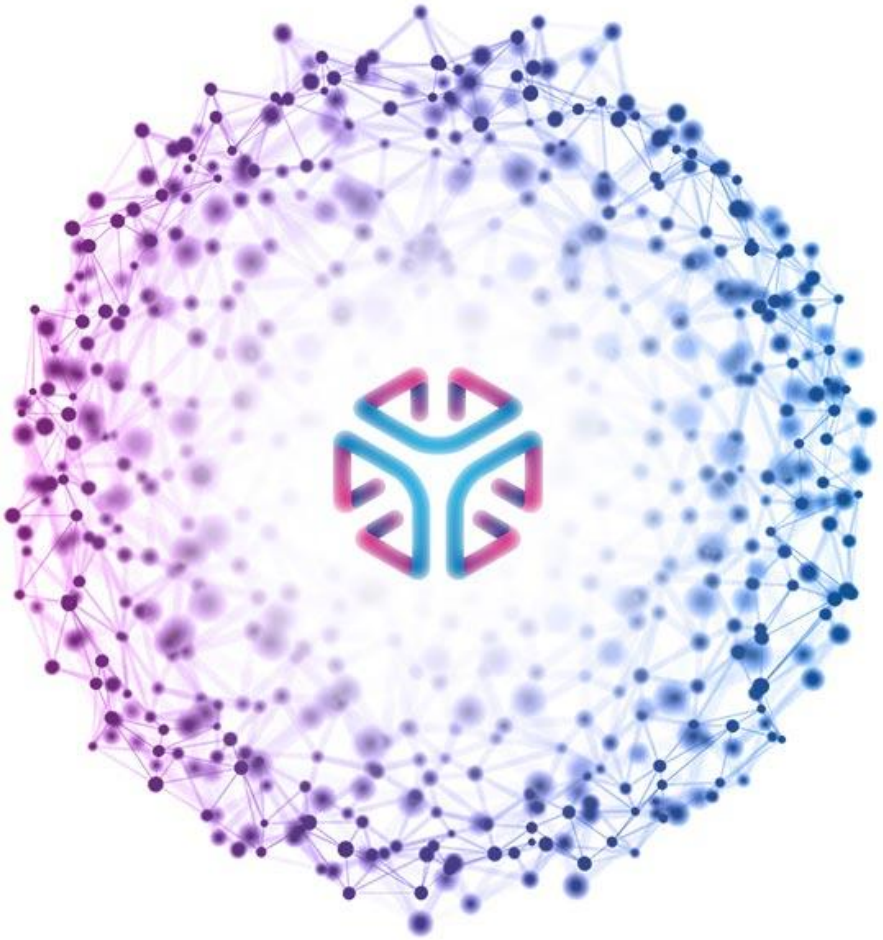
The Nahmii project began in 2017, when Norwegian startup Nahmii AS decided to move its existing content aggregation business to the blockchain. Although Ethereum was the obvious choice for a platform to build on, we immediately understood that a scaling solution would be required. Ethereum alone could not meet our performance needs. After careful evaluation of the solutions in development at the time, it became clear that no proposed Ethereum scaling solution could deliver what we required: a protocol capable of handling millions of transactions per second, with predictable low fees, minimal latency and instant transaction finality.

With no workable solution available, we decided to build our own. In early 2018, Nahmii AS began work on the Nahmii project; an Ethereum-based scaling solution capable of handling commercial-scale blockchain applications.

Development of the protocol moved swiftly throughout mid-2018, with the original version of Nahmii's white paper published in June 2018. At the same time, Nahmii's smart contracts were deployed to the Ropsten test network. Following extensive testing, Nahmii was deployed to the main Ethereum network in four stages starting in November 2018. The protocol has been fully live since March 2019 and includes the ability to make deposits, send payments, settle accounts and withdraw. Despite starting long after our competitors, we succeeded in delivering the first ever commercially-viable scaling solution for blockchain.

Now, following significant further development of the protocol, we have chosen to update Nahmii's white paper to reflect the progress made and coming updates. As with the original document, the purpose of this white paper is to offer an approachable insight into the Nahmii protocol. Our aim is to use non-technical language where possible, although some basic knowledge of blockchains and Ethereum is required to understand everything. Similarly, the examples set out in this paper - notably relating to fraud detection - do not cover all eventualities.

All of Nahmii's smart contracts are public, as are many of Nahmii AS's GitHub repositories relating to the protocol and its development tools. Given the public nature of this code, the Nahmii code itself represents the ultimate technical documentation. Despite this, we will maintain documentation which simplifies the principles behind Nahmii for a wider audience.



Introducing Nahmii 2.0

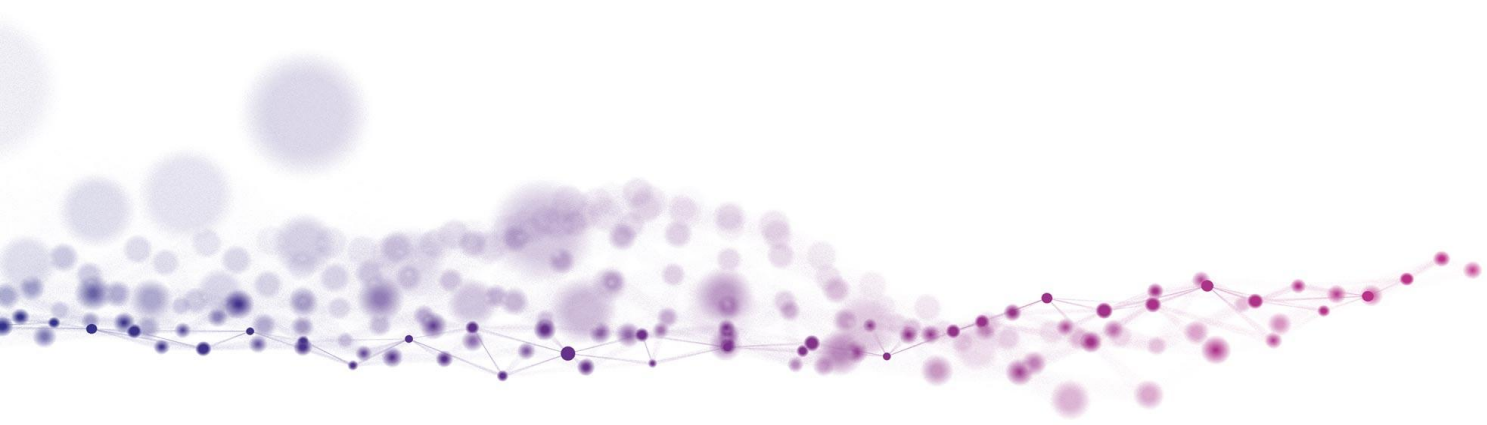
Nahmii 2.0 is a comprehensive upgrade to Nahmii, which will enable smart contract capabilities, whilst preserving the key attributes of the original protocol. Only Nahmii 2.0 combines smart contracts and commercial scaling performance in this way, backed by our unique patent-pending architecture.

We began building Nahmii in 2017 after identifying many issues with Plasma-based scaling solutions, which would ultimately prevent Plasma achieving any level of adoption. Rollups and side-chains of various types are mostly evolutions of the principles behind Plasma and, as a result, inherit many of the same performance constraints and security problems. These issues mean that products at scale cannot be built upon these platforms. Nahmii represented a paradigm shift for scaling payments in 2018 and Nahmii 2.0 is the same for generalised computation today.

This difference is explained by the fundamental architecture underpinning each approach: Nahmii does not require regular commitments to the slow base-layer blockchain, unlike most other solutions. We still rely on Ethereum for Nahmii's security, but - importantly - not before a transaction takes place. We only use Ethereum for security ex post facto; a critical and unique method to ensure that users' funds are always safe whilst allowing unparalleled scaling.

Any approach that uses the base layer for periodic commitments to enact transactions is limited to the latency and finality of the underlying blockchain. This represents a significant performance handicap. Instead, Nahmii avoids the need for such commitments to Ethereum, bypassing the performance issues associated with this technique. It is not trivial to inherit the security of Ethereum without making such commitments, but Nahmii's patent-pending methodology made this possible for the first time.

Our work since 2018 has demonstrated that there are no technical barriers to achieving security comparable to the base layer of Ethereum in a live layer-2 protocol, unlike many other solutions which are still at a research stage today.



Governance and Security Model

Nahmii will be governed in accordance with a hybrid foundation and governance token model. The token (NII) is not only used for governance but is also key to the security model of Nahmii.

All transaction fees generated by users of the protocol will serve the security of Nahmii. The overwhelming majority of transaction fees will be earned by NII token holders in return for their participation in the security mechanisms of Nahmii. This ensures that those token holders are economically bonded to Nahmii and this is critical to strongly incentivise the protection of the protocol. It would be impossible to build Nahmii without such a direct incentive structure.

All entities are responsible for the efficient and reasonable management of the protocol. Token holders represent a distributed body of vested entities, whereas the Nahmii Foundation would be a legal organisation. In the Nahmii Foundation, all members, Nahmii AS included, will be equal partners with the same associated rights, privileges and responsibilities. Members will be geographically distributed and leading companies across diverse industries, ensuring that the Foundation itself will be decentralised in nature.

There will be some additional decisions which the Foundation is best placed to make. Whilst some governance is best served by a network of token holders; the Foundation will not make decisions which should be made in a more distributed fashion.

A fraction of the transaction fees generated by Nahmii will be used to buy NII using an appropriate decentralised exchange. This buyback will be managed by a smart contract. This NII will be used exclusively by the Foundation in the Data Availability Oracle. The fees collected by those NII tokens will be controlled by the Foundation and used for the benefit of the protocol, including funding

products built on Nahmii. The amount of tokens that the Foundation can accrue should be capped and this is an example of governance which will be performed by NII token holders and not the Foundation itself.

Members of the Nahmii Foundation are required to play an active role in monitoring, validating and protecting the protocol. Their role also includes a commitment to building Nahmii-based solutions and the provision of nodes to assist protocol operation. The Foundation might be required to ensure the responsible divestment of Nahmii's large holding at appropriate points in time. This divestment is part of a broader plan to guarantee a plurality of Nahmii token holders, thus making the protocol more resistant to a 51% attack.

Over time the intention is that the boundaries between the Nahmii Foundation and token holders will become increasingly blurred. Eventually the governance of Nahmii should be as close as possible to a Decentralised Autonomous Organization (DAO).

Key Requirements

Trustlessness

Nahmii employs an Operator to countersign transaction requests. The key design principle behind Nahmii is that it does not rely on users trusting the Operator; the system has been designed and is required to have a clear path to trustlessness. If the Operator is malicious or is compromised, it will not be possible for them to defraud users.

A trustless protocol is one where there is no requirement to trust the Operator. All transactions using Nahmii will be processed initially by Nahmii AS, who will act as the Operator of the protocol. Later on this will be decentralised across multiple Operators. At all times the Operator (and other users where appropriate) must be held accountable for their actions, preventing fraud and thereby protecting users' funds. It is also essential that users must be able to exit from Nahmii even if the Operator goes offline. This is not currently the case for some other popular layer 2s. For Nahmii it is.

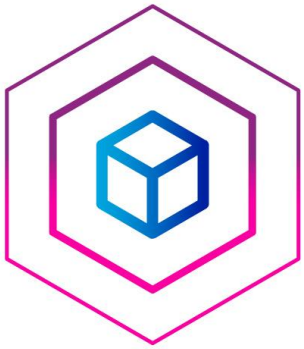
It is important to state that the protocol will not be fully trustless at first release, but will instead move towards trustlessness in an iterative fashion. This is similar to other proposed layer 2s, which impose varying trust requirements initially. The reason behind Nahmii's staged process does contrast with other layer 2s, because others have not yet addressed a variety of vulnerabilities or have outstanding research requirements. Nahmii has the most clear and well defined path to complete trustlessness and could, in fact, be trustless immediately. The simple explanation for not doing so is that we will not compromise on delivering value; it is of paramount importance that a piece of infrastructure is able to react and change to user and product requirements. The established speed of development in the blockchain space is both too slow and insufficiently agile for products targeted at billions of users.

This is not viewed as a barrier to adoption, as there are no trustless alternative layer 2s on the market. There are also no alternatives with a clearer path to full trustlessness than Nahmii. Furthermore, we have seen that the majority of users still prefer to use high-performing products, e.g. centralised exchanges, over those which are lower performance but trustless. Nahmii, backed by Ethereum, will deliver against all the promises of blockchain technology.

Documentation will always be made available allowing users to easily view the state of the network in terms of trustlessness. There will be no obfuscation of which parts of Nahmii are not yet trustless, unlike other layer 2s.

Many aspects of Nahmii will be decentralised immediately, but the protocol is not designed to require full decentralisation. Eventually, Nahmii will be fully decentralised, but this is not the immediate focus. In many cases decentralisation can cause needless compromises. A separation of concerns is a well-known software design principle which we have applied to the architecture of Nahmii, in order to achieve the requirements listed here.

Censorship resistance or being permissionless are not the focus of Nahmii. Whilst it is possible to enable these 'features', and in some cases we will do, this is not desirable for many of the markets and customers we are targeting. As an example, for a wide range of financial use cases, it is essential to be able to comply with regulations and enact Know Your Customer (KYC), Know Your Transaction (KYT) and Anti-Money Laundering (AML) rules. The ability to apply such controls is seen as a real 'feature' of Nahmii by many potential customers. This will not prevent us from simultaneously implementing use cases where censorship resistance or being permissionless is desirable. Our goal is for walled gardens to be able to coexist with open innovation in the Nahmii ecosystem.



Scalability

Since the dawn of blockchain there has been constant talk about scalability. This discussion has centred on the concept of Transactions Per Second (TPS). This is understandable, as TPS as a metric is both important and easy to compare across solutions, but it is not the only requirement for true scalability. Since 2017, we have gone to great lengths to describe the problems of taking a narrow viewpoint around scaling. Nahmii has always been the only holistically-designed layer 2, focused on optimising additional key performance metrics including low latency, instant finality and predictability alongside TPS. This principle will continue to guide us into the future.

High Throughput

We do not deny the critical importance of maintaining high throughput or TPS for almost every possible blockchain use case and we understand why it has become the primary talking point around scalability. The requirement for Nahmii is that the throughput of Nahmii is *horizontally* scalable. In this context, this means that Nahmii is able to process transactions at a speed on par with traditional database systems - far beyond any other layer 2.

In addition to horizontal scaling, we facilitate many transactions per second for each individual wallet. This is not always the case for other layer 2s and we consider it to be an overlooked requirement.

Low Latency

The latency of a typical blockchain transaction is extremely high, yet we live in a world of global products where great resources are expended on reducing latency. For an Ethereum transaction, we define the time between submitting a transaction and first confirmation on the network as latency. Assuming a sufficiently high gas price is paid for an Ethereum transaction, the average latency is at least half the average block time, or approximately 6.5 seconds. This is unsuitable for all but the most niche products at scale.

Furthermore, for almost all other layer 2s, latency is actually inferior to Ethereum itself. In some cases it is significantly worse, ranging from several minutes to hours. There is no way to fix this issue for

those other layer 2s and suggestions of economic bonding are impractical. Nahmii does not rely on such false promises of transactions having taken place.

As noted by Vitalik Buterin in the passage below, bonded promises to include a transaction in the next block are insufficient to guarantee the security of such a system:

"Many rollups provide a notion of "pre-confirmation" for faster UX, where the sequencer immediately provides a promise that a transaction will be included in the next batch, and the sequencer's deposit is destroyed if they break their word. But the economy security of this scheme is limited, because of the possibility of making many promises to very many actors at the same time." - Vitalik Buterin, "An Incomplete Guide to Rollups", January 2021.

Nahmii's latency is defined as the time from submitting a transaction request to receiving a transaction receipt. Excluding any latency caused by a user's connection stability or speed, this is measurable in milliseconds.

Instant Finality

The finality of a transaction is the point past which it cannot be, or is extremely unlikely to be, reversed. Outside of the blockchain space, transaction finality can be minutes, days or even months; businesses use various methods to protect themselves from transaction rollbacks caused by fraud or other reasons. Ethereum has a significant advantage here, in that finality is achieved within minutes. This opens up huge opportunities, but can also cause problems if there is uncertainty.

An example of this is in the emergence of Decentralised Finance (DeFi). DeFi has been a phenomenal success in many ways, but there are risks associated with and created by the time taken for a transaction to be finalised. Taking a simple example of trading, it is apparent that if a transaction has a finality of several minutes, this delay creates risk for arbitrageurs, market makers and high-frequency traders. Consequently the efficiency and liquidity of DeFi markets is reduced.

The majority of layer-2 scaling solutions have finality which is intrinsically linked to that of Ethereum. Ethereum 1.0 has a probabilistic finality of around 2-3 minutes. Ethereum 2.0 will have an absolute finality of approximately 13 minutes. As described above for latency, claims of utilising bonds to ensure economic finality on other layer 2s are misleading.

Nahmii is required to have instant finality. Once a transaction receipt is observed and validated, it can be considered final. That transaction is irreversible and this is akin to the transaction finality of state channels.

Predictable Transaction Fees

Anyone who has used Ethereum will be familiar with how variable transaction fees can be. Although each transaction uses the same amount of 'gas', there is an additional 'gas price' parameter, which allows users to increase the bid for their transaction to be included by miners. This gas price has varied over at least five orders of magnitude in the time that Ethereum has existed. Equally significantly, prices can vary hugely from day to day. It is simply not possible to build a product for the masses with this level of price variability.

For commercial adoption, Nahmii will ensure predictable pricing. This is a direct result of Nahmii's horizontal scaling in terms of TPS.

Low Transaction Fees

Related to the requirement for predictable fees, it should also be apparent that low fees are essential. Nahmii supports fees that enable transactions of low value.

Generalised Smart Contract Support

The primary upgrade in capability between Nahmii 1.0 and Nahmii 2.0 is the ability to support generalised smart contracts. For Nahmii 1.0, there was only support for payments, although there was the ability to manually add custom transaction types. We deemed this inefficient and shifted the focus to further developing Nahmii to be able to support any smart contract.

Composability

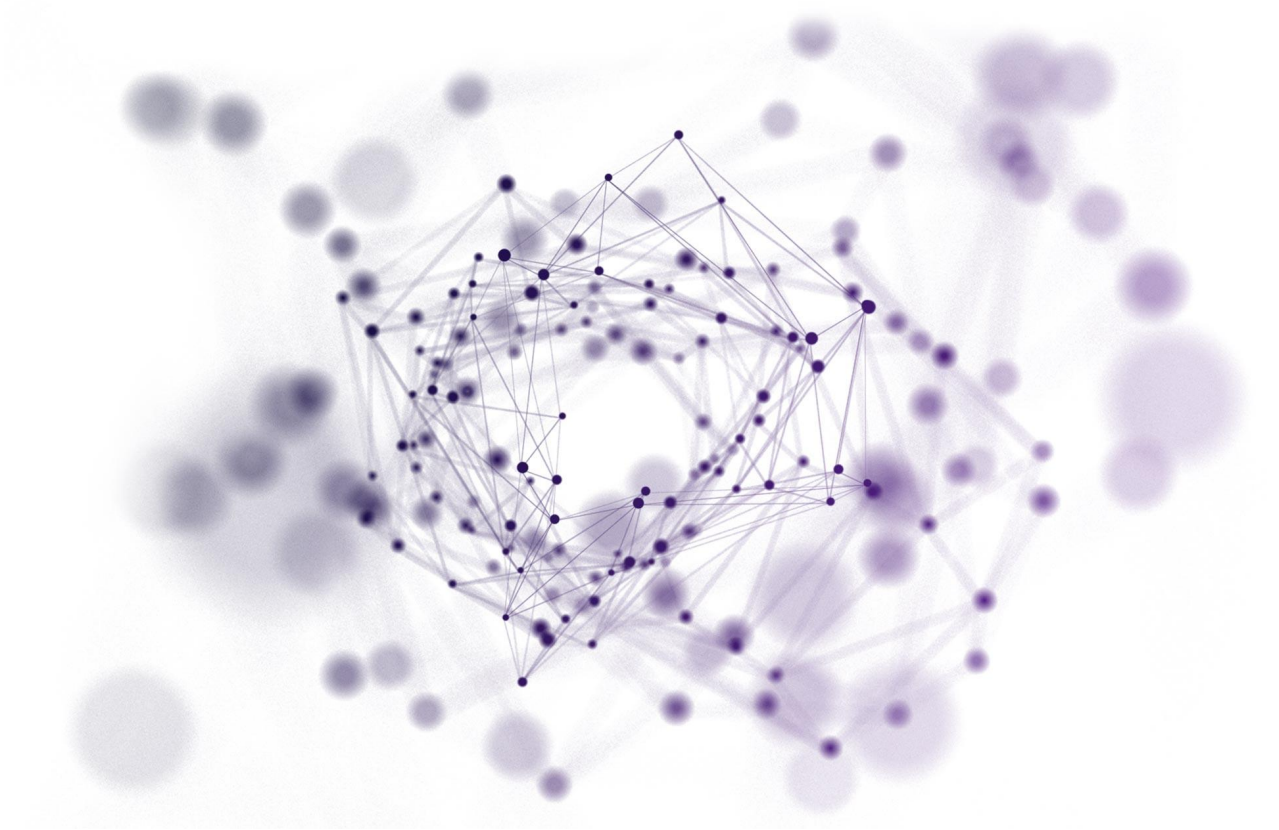
Much of the recent success of DeFi can be attributed to the ability to have smart contracts interact with each other. This has been occasionally termed 'money Lego', but is more accurately called 'composability'. Composability allows components of a system to be assembled in combinations to create a desired result. Some layer 2s support composability, whilst others do not. We feel it is critical for Nahmii to allow full composability in order to build powerful new products.

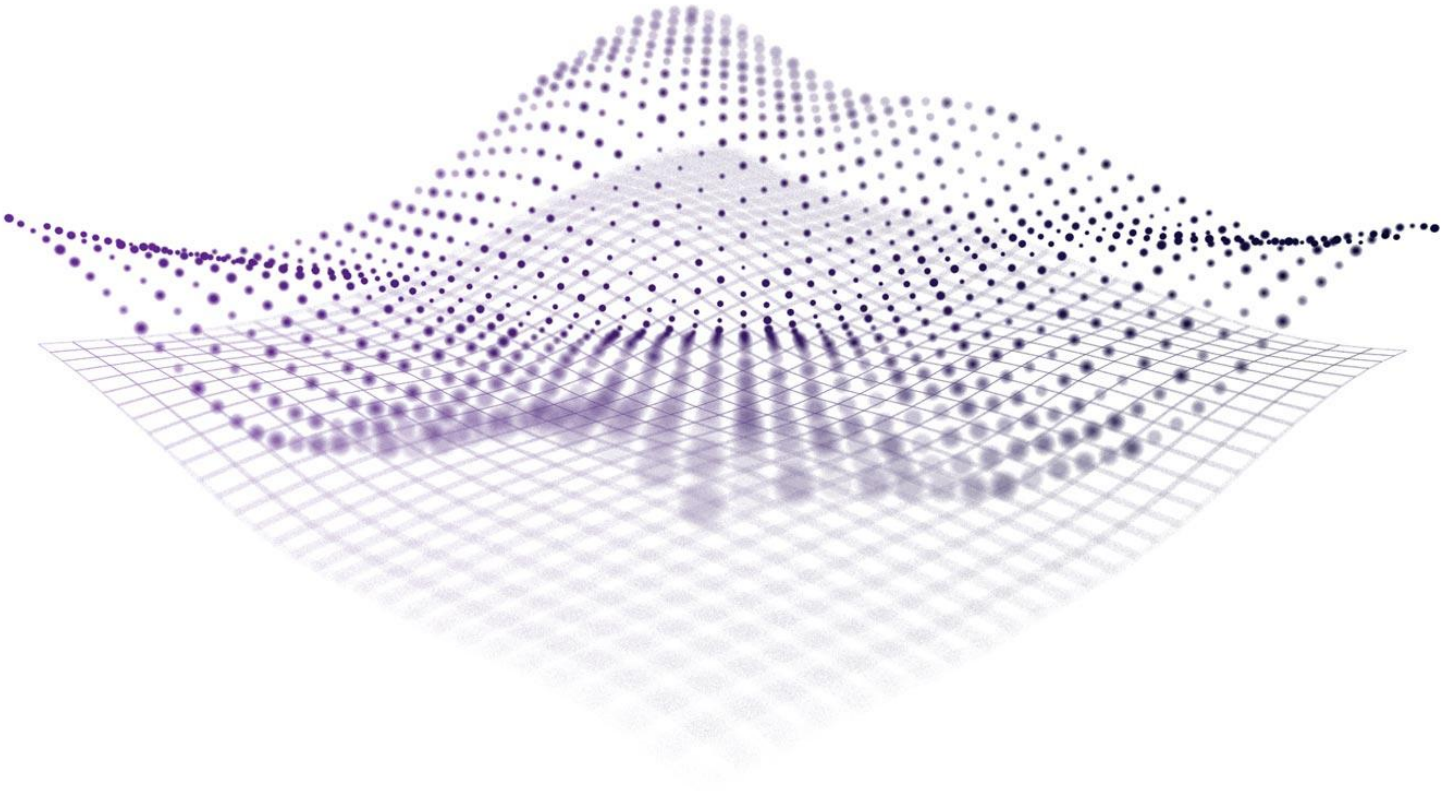
Privacy

Privacy is a highly desirable feature for many mainstream use cases of blockchain technology. For example, a company might want to use a blockchain for supply chain finance, but not reveal too much information about pricing to their competitors. Enabling generalised smart contract support is a key tool to allow such important privacy features to be enabled.

Ethereum Tooling Compatibility

It is important to maintain as much compatibility as possible between existing Ethereum tooling and Nahmii. This minimises the amount of work to be done for Nahmii to take advantage of the vast and diverse talent that has already moved into the Ethereum ecosystem. It also ensures that there is plentiful documentation already available for developers that are only now moving into the space.





Architecture

State Pools

We use the term ‘state pool’ to describe the fundamental architecture behind Nahmii. This name reflects the close relationship between Nahmii and state channel-based approaches, hence ‘state’. Similarly, Nahmii utilises similar pooled security models from side-chain systems, such as Plasma, Rollups etc., hence *‘state pool’*.

This hybrid ‘state pool’ model is unique and allows Nahmii to deliver unrivalled scaling performance. In this document we will explain how Nahmii is able to inherit the good qualities of both state channels and side chains, whilst avoiding the bad.

Security Model

The Nahmii security architecture is divided into three levels, each containing a set of participant nodes. This separation of concerns is best understood as:

Operator

The Nahmii protocol will be operated initially by Nahmii AS, who will provide the first point of validation on the platform. This arrangement is subject to the governance and approval of the Nahmii Foundation. Much of Nahmii's security construction is designed to ensure the Operator is unable to commit fraud, even if compromised. Over time, there will be multiple other Operators added to the network.

Transaction Monitoring

Transactions within Nahmii generate receipts, which can be used to prove user or Operator fraud. If a user attempts to settle with an old and invalidated state, this can be challenged by submitting a fraud proof to Nahmii's smart contract. The reward for challenging a settlement in this way is the user's Settlement Bond, which is only posted if a user wishes to settle their state back to the layer 1. Additionally, a transaction can be challenged at all times if it exhibits Operator fraud. The reward for a successful Operator fraud challenge is a share of an Operator Bond. Any Ethereum user can take part in this system of fraud proofs.

These challenges are explained in more detail in the 'Continuous Fraud Challenge' and 'Settlement Challenge' sections. The Operator Bond and Settlement Bond are described in the 'Operator Bond' and 'Bonded Settlement' sections respectively.

Data Availability Validation

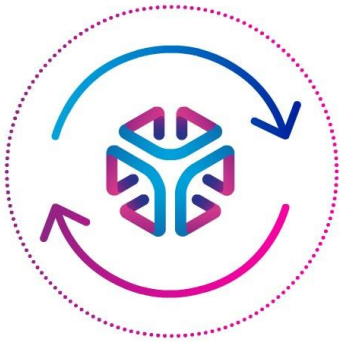
The fraud challenges detailed above require users to submit proof in the form of transaction receipts. These receipts are published by the Operator and Nahmii token holders are responsible for validating that this data is available at all times. The 'Data Availability Validation' section provides more detail on this system.

The security provisions within the Nahmii protocol are designed to protect against three possible fraudulent attacks, characterised in terms of data availability. First, users are protected against the possibility of a compromised Operator through the 'Continuous Fraud Challenge' mechanism. Second, users are protected against illegitimate rollbacks through the 'Settlement Challenge' mechanism. These challenges require transaction data (in the form of receipts) to be both available and accurate, hence the need for a third security provision for when data is not available. A fully decentralised 'Data Availability Oracle' was proposed, designed and tested as part of Nahmii 1.0. The Oracle will continually monitor data availability; this is the third security provision.

The Nahmii security model relies on a system of rewards and punishments, guided by the general principle that there should always be a sufficient incentive to encourage good behaviour. Any Operator of Nahmii will have a clear stake in the continued success of the protocol, primarily because of the bond they must post in order to run the system. The same is true for Nahmii

Foundation members and NII token holders, who share a financial interest in the safe operation of Nahmii.

The following section sets out the three security provisions in detail, explaining the rationale behind each and how they work together to ensure the safe operation of the Nahmii protocol.

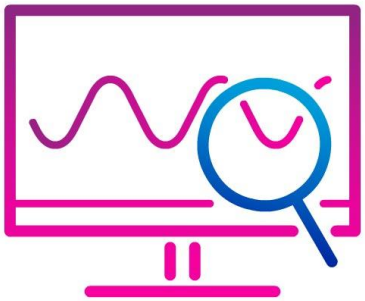


Operator

The Operator of the Nahmii protocol will be Nahmii AS initially, with the option to decentralise this processing later with the support of the Nahmii Foundation. Transactions within Nahmii require the countersignature of the Operator, who should only allow valid state transitions to proceed.

The security constructions within Nahmii have been designed to protect user's funds in the event of a rogue or compromised Operator. In the event that the protocol has been compromised, Nahmii can continue to operate to ensure commercial applications have no operational issues. The Operator will be suitably punished for this issue and this is described in the 'Operator Bond' section of this document.

The security of Nahmii is further enhanced by the foundation model of governance, which makes the possibility of a malevolent Operator gaining overall control substantially less likely.



Transaction Monitoring

Continuous Fraud Challenge

In order for Nahmii to approve a potential transaction, it must first be signed by both the user initiating the request and the Operator confirming it. The integrity of the protocol depends on the Operator only signing valid transactions, hence the requirement for a security check to ensure that this is the case. This 'Continuous Fraud Challenge' is therefore designed to protect users from the possibility of a compromised or rogue Operator. Importantly, users are required to check their transaction ancestry before accepting it. This is a trivial process using software, as explained in further detail later in this document. This gives them certainty that the transaction won't be reversed, hence why Nahmii has instant finality.

Below, we provide a simple example of how this challenge can work but it is important to note that the Nahmii protocol must be able to detect all forms of fraudulent state changes.

The twin sign-off process is best illustrated by way of a simple example, a request by Alice (A) to make a transfer of 100 tokens to Bob (B). First, Alice initiates the transfer request and signs it with her private key to verify the transaction. Next, the Operator (O) checks this is a valid state change.
Provided that it is valid, the Operator countersigns the transaction. After performing this second check, Alice's balance is decreased by 100 tokens and Bob's balance is increased by the same amount. Finally, the Operator reveals the transaction publicly in the form of a standardised receipt. The process can therefore be represented as:

1. A initiates a transfer request to send 100 tokens to B
2. A signs the transaction using her private key
3. O checks that the transaction represents a valid state change (it does)
4. O countersigns the transaction
5. A's balance is decreased by 100 tokens
6. B's balance is increased by 100 tokens
7. O publishes the transaction receipt

This example assumes that the Operator has not been compromised. If the Operator has been compromised, there are two possible ways in which Operator fraud can occur. In both cases, the transaction receipt provides sufficient evidence to prove Operator fraud when submitted to Nahmii's security smart contracts.

The first possible Operator fraud is easy to spot, as it involves the Operator approving an invalid state change. If Alice initiates an invalid state change, such as a transfer for more than her available balance of tokens, the Operator should not countersign the transaction. We cannot say for certain whether this is attempted fraud on Alice's behalf as it could be explained by innocent user error or malfunctioning software, but it is *always* viewed as Operator fraud. This is best illustrated by way of an example:

1. A initiates a transfer to send 100 tokens to B (her balance is 0 tokens)
2. A signs the transaction using her private key
3. O checks that the transaction represents a valid state change (it does not)
4. O countersigns the transaction
5. In this example of an invalid state change, A's balance is not decreased by 100 tokens
6. B's balance is increased by 100 tokens
7. O publishes the transaction receipt

The reason why Operator fraud has been committed in this example is because Alice's invalid state change has been approved by the Operator. Alice does not have 100 tokens in her balance to send to Bob, so the Operator should not have countersigned this transaction.

In the second Operator fraud scenario, the Operator changes the transaction byte code before approving the modified transaction. A simple example would be a transfer between Alice and Bob, which has been signed by Alice. A compromised Operator could change the transfer amount or payee address, then approve this fraudulent transaction. Clearly, Alice is not at fault here; it is trivial to prove that Alice did not sign the modified transaction. This is, however, a clear case of Operator fraud. Once again, this is best explained through an example:

1. A initiates a transfer request to send 100 tokens to B
2. A signs the transaction using her private key
3. O checks that the transaction represents a valid state change (it does)
4. O changes the details of the transaction to pay Carol (C) instead
5. O countersigns the modified transaction
6. A's balance is decreased by 100 tokens
7. C's balance is increased by 100 tokens
8. O publishes the transaction receipt

The fraud that has been committed in this example is that the recipient of the tokens is now Carol instead of Bob. This was not the intention of Alice when she initiated the transaction. It is important to note that there is no evidence that suggests Carol has done anything wrong.

Both types of Operator fraud are revealed by examining the relevant transaction receipts. In the first example, the before and after balances will not reconcile. For the second, Alice's signature will no longer match the transaction. In each case, the transaction receipt is sufficient to prove Operator fraud has taken place. These receipts are publicly available and any user can submit them to Nahmii's security smart contracts as a fraud proof.

Importantly, the beneficiaries of Operator fraud can never settle and withdraw funds from Nahmii based on their invalid states. There are two methods by which users are protected from fraudulent state transitions. First, fraudulent transactions can be challenged by any user before they are used as part of the settlement and withdrawal process. This is the first application of the 'Continuous Fraud Challenge' mechanism and can take place at any time. Secondly, if a user attempts to settle and withdraw based on a fraudulent transaction, the user must reveal the fraudulent transaction receipt as part of the settlement process.[2] This receipt can now be used to prove Operator fraud. Even though this fraud proof will take place during a settlement attempt, it is still considered to be part of the 'Continuous Fraud Challenge'.

The 'Continuous Fraud Challenge' mechanism protects Nahmii users by identifying fraudulent transactions before they are used to settle and withdraw, but this alone does not mitigate against all possible risks. There is a requirement for all users to check their transaction ancestry, i.e. that their transactions are not built upon obviously fraudulent foundations. In our last example, Carol receives funds that were intended for Bob. If Carol had tried to spend those funds, it is the responsibility of the payee to check whether the transaction is based on fraud before accepting it as legitimate. Thankfully, provided that all data is available, this check is trivial when using software.[3] Once the payee has validated the transaction, there is no possibility of it being reversed, hence Nahmii has instant finality.

The key to identifying fraudulent transactions of this type lies in step 8, where the Operator publishes all transaction data. Based on this information, users of the Ethereum network can challenge a transaction by submitting a fraud proof. All of the necessary information is publicly available to prove the fraud. Once a successful proof is submitted to the smart contract, the Operator is punished and the user who submitted the fraud proof receives a reward from the Operator Bond.

---

[2] Or that there is fraud or missing data in the transaction ancestry pertinent to their settlement. The handling of missing data is described in the 'Data Availability Validation' section.

[3] The handling of missing data is described in the 'Data Availability Validation' section.

An example of a successful challenge by Dan (D), representing any person who has become aware of the fraudulent transaction, is:

1. A initiates a transfer request to send 100 tokens to B
2. A signs the transaction using her private key
3. O checks that the transaction represents a valid state change (it does)
4. O changes the details of the transaction to pay Carol (C) instead
5. O countersigns the modified transaction
6. A's balance is decreased by 100 tokens
7. C's balance is increased by 100 tokens[4]
8. O publishes the transaction receipt
9. D raises a challenge against the state change by calling the appropriate smart contract
10. D provides a fraud proof using the data from step 8
11. Smart contract confirms fraud and will punish the Operator
12. D is rewarded from the Operator Bond

The fraud proof process is not free, as calling the smart contract incurs a gas cost for Dan. This necessitates that the reward from the Operator Bond is a sufficient incentive at all times to overcome this cost. This widens the set of users incentivised to submit a fraud proof to any user of Ethereum, instead of it being restricted to Nahmii token holders who are economically bonded to the value of the network.

We anticipate that many nodes will be monitoring the published transaction data for anomalies, including nodes run by NII token holders who are doubly incentivised to secure the network. In addition, there will be nodes which are run by the Nahmii Foundation partners. Please refer to the earlier 'Governance and Security Model' section of this document for more information regarding these points.

Settlement Challenge - Basic Principles

A key part of any layer-2 protocol is settlement, which explains how transaction data from the layer 2 is reconciled with the underlying blockchain. For Nahmii, settlement is effectively the method by which a user's state on the layer 2 is reconciled with Ethereum.

The Nahmii settlement process requires the settlement of a general state, which covers a much broader range of possibilities than the equivalent settlement process in Nahmii 1.0. Settlement is no longer simply a matter of reconciling token balances before withdrawal, instead it refers to the alignment of state across both layers.

[4] No party in the network, including C's software, would accept this fraudulent transaction or any transaction built upon it.

In order to allow a consensus to form around whether a settlement request is valid, the Nahmii settlement process includes a 'Challenge Period'. Once a settlement begins, a timer starts and any user can submit evidence to Nahmii's smart contracts that the settlement is fraudulent during this window. Transaction receipts, published by the Operator, are used as fraud proofs; this explains why data availability is so important to the security of the protocol. If the Challenge Period ends without a successful settlement challenge being raised, the user who started the settlement can then exit Nahmii safely. An example of a non-fraudulent settlement process is below:

1. A deposits 100 tokens into Nahmii (A's balance is 100 tokens)
2. A transfers 50 tokens to B (A's balance is now 50 tokens)
3. A starts a settlement and provides the transaction data from step 2 as evidence
4. The Challenge Period begins
5. The Challenge Period ends
6. No settlement challenge has been raised against A's settlement request during the Challenge Period
7. A's layer-2 state is now considered settled (i.e. reconciled) with her layer-1 state

To ensure that both users and Nahmii's Operator are protected, settlement requires the user to post a 'Settlement Bond'. This bond is at risk in the event of a fraudulent settlement being detected, where this applies only when the user could be at fault. The most common method by which a user could attempt to defraud Nahmii is by trying to settle their wallet based on an old state, which is illustrated below:

1. A deposits 100 tokens into Nahmii (A's balance is 100 tokens)
2. A transfers 50 tokens to B (A's balance is now 50 tokens)
3. A transfers 50 tokens to C (A's balance is now 0 tokens)
4. A starts a settlement and provides the transaction data from step 2 as evidence

In this case, Alice is attempting a fraudulent early settlement. By using the transfer receipt from step 2 as evidence, Alice is trying to settle a state where her wallet balance is 50 tokens. As we know, Alice has actually transferred those tokens in step 3, hence the attempted fraud. Her real balance is 0 tokens. Crucially, the smart contracts underpinning Nahmii on Ethereum do not have access to the transaction receipts. It is only when the smart contract is presented with the second transfer receipt as a fraud proof that Alice's fraud is formally detected.

An example of a successful settlement challenge by Dan (D) is shown below, which includes Alice losing her Settlement Bond:

1. A deposits 100 tokens into Nahmii (A's balance is 100 tokens)
2. A transfers 50 tokens to B (A's balance is now 50 tokens)
3. A transfers 50 tokens to C (A's balance is now 0 tokens)
4. A starts a settlement and provides the transaction data from step 2 as evidence. A posts a Settlement Bond
5. The Challenge Period begins
6. D raises a settlement challenge against A's settlement by providing the receipt from step 3 as evidence
7. Nahmii's smart contract checks the evidence and confirms the settlement is fraudulent
8. A's settlement does not proceed
9. D is rewarded with A's Settlement Bond

These examples outline the basic process by which Nahmii settlement functions, which is sufficient to explain the key principles. A more technical overview is provided in the next section.

Settlement Challenge - Technical Details

A user will generate a settlement transaction for the L1, but this must be reflected in the L2 state also. Ordinarily a user will do this settlement by sending the L1 transaction to the Ethereum network via the Operator's nodes. This way, the Operator can ensure that this settlement is reflected in the L2 state in case the user signs another transaction immediately. The Operator needs to lock the user out of transacting (perhaps only on an individual contract basis) until the settlement transaction is mined on Ethereum and is x blocks deep (or the epoch is finalised for Ethereum 2.0). Once the transaction is finalised then the Operator can allow the user to transact again with a state that accurately reflects the settlement.

All products should use the same API, to avoid any conflict in state. All products should also take care of the user as per the below.

It is critical for Nahmii to have a trustless settlement method, which does not involve the Operator. Users must be able to initiate a settlement without the Operator's permission. When a user does so, then they must refrain from signing any further transactions that would conflict with their settlement state, or they will risk being punished for having an 'overrunning' state change. It is safest for them to not perform any transactions until their own settlement has been confirmed, even if they are able to check that a subsequent transaction request is correctly reflecting their state post-settlement.[5]

[5] Depending on exact implementation, we may need the ability for users or their software to request the Operator signs to acknowledge the settlement after x confirmations (or the epoch is finalised for Ethereum 2.0). This would ensure that a user is protected from a malicious Operator that is trying to put them into a punishable position. If the Operator refused to sign this acknowledgement then the user would exit the system and perform no more transactions in Nahmii.

It is not always the case that an Operator would be to blame for not accurately reflecting state, as a malicious user might use the trustless method of settlement and then sign transaction requests that do not take into account this settlement. We can imagine a situation where the Operator signs a state which conflicts with a settlement attempt because they haven't seen it yet (perhaps the settlement attempt hits the Ethereum mempool at the same time as Alice requests to perform a transaction). The Operator is innocent but Alice should be punished even though she, in theory, might also be innocent.

Alice may be innocent, and this problem could be caused by badly implemented software, but the onus is on protecting the network; we must assume the individual is guilty. Users would ideally use well-tested, open-source software to interact with Nahmii. Implementers of Nahmii integrations must take responsibility for poor software which leads to user's losing funds.

As such, settlement on Nahmii should be considered an expert feature. We do not anticipate that a regular user will perform settlements.

Data Availability Validation

The Data Availability Problem

The fraud and settlement challenges set out earlier in this section rest on the principle of data availability, which requires that the Operator publish accurate and complete transaction receipts at all times. This data is crucially important for ensuring confidence in Nahmii and the Operator. Without the relevant data, users cannot challenge fraudulent transactions as there is no evidence to send to the smart contract. Similarly, the off-chain elements of Nahmii cannot be verified without the transaction receipts; users cannot therefore be confident that their transactions have been processed correctly by the Operator without access to the appropriate data.

Nahmii users cannot simply presume that the Operator is trustworthy, they must be able to inspect the published data to be sure. As such, Nahmii requires a decentralised method by which the smart contract can check that data is available. Furthermore, this must be secured against external

manipulation. This is an extremely difficult problem to solve and the most challenging part of building a layer-2 protocol.

The challenge described above is known as the 'data availability problem'. If users of a protocol like Nahmii need the Operator to publish transaction data to check whether the Operator is trustworthy, how can users ensure that the Operator is publishing the appropriate data? Our solution is the 'Data Availability Oracle', a fully decentralised mechanism by which distributed Nahmii token holders are incentivised to monitor data availability.

Data Availability Oracle

The 'Data Availability Oracle' is designed to protect Nahmii users from a potentially compromised Operator who is withholding data. As discussed, the 'Continuous Fraud Challenge' and 'Settlement Challenge' processes rely on users submitting proof of a fraudulent transaction. This proof is taken from data published by the Operator, without which the challenges cannot function. In the unlikely event that a compromised Operator chooses to publish fraudulent transaction data, this attempted fraud is easily identifiable. Far more likely is the alternative scenario, whereby a compromised Operator refuses to publish the relevant data (either by selectively excluding certain transactions or simply withholding all data). In this case, the proof of the attempted fraud is the *absence* of data rather than the presence of tangible evidence. This requires a different kind of solution.

The Oracle is best understood as a function of the account settlement process within Nahmii. Before a user can withdraw funds, they must first settle their account. The first step in the withdrawal process is therefore to check whether the user's transactions for the settlement are all valid, this requires the user to call a smart contract and start the Challenge Period. During this period, any Ethereum user may challenge the settlement by providing evidence in the form of a transaction receipt. If the Challenge Period ends without a successful challenge, the user can then effectively reactivate the smart contract (which has been dormant during the Challenge Period). As the settlement request has not been proven fraudulent, the smart contract performs one final check before finalising the settlement: *is data available*? This is done by querying the Oracle.

In order for the Oracle to function, it must return a binary response when the smart contract checks whether data is available (yes/no, true/false etc.). The Oracle is a game theory-based distributed intelligence tool, which has been in development by Nahmii AS since 2017. It operates on the principle of a small reward for being a good actor and a severe punishment for being a bad actor. The purpose of the Oracle is to continually test several statements relating to data availability during a settlement Challenge Period. These statements have only two truth conditions, true or false (or equivalents, i.e. no indeterminacy is possible), where the combination of these statements provides a similarly clear answer to the question 'is data available?'. The outputs of the Oracle should also account for the possibility of a legitimate, temporary problem with data availability

which is fixed later. If the Oracle subsequently returns the response indicating that data is available again, the Nahmii settlement process is effectively reactivated.

Questions around data availability must necessarily include a temporal component, as data availability can change over time. This relates to the staking mechanism at the heart of the Nahmii Oracle, whereby Nahmii token holders stake their tokens against 'true' or 'false' for each of the data availability statements. It is trivial to see why the monitoring market must include a temporal component. Without this time restriction, two users may stake their tokens on opposite outcomes and both be correct. Consider the simple statement 'data is available', this can be both true at time t_1 and false at time t_2 within the settlement period. It is therefore essential that the crucial test statements are formulated correctly. In order for the status of a question to change between 'true' and 'false', then the staking metrics must meet a variety of criteria. It is insufficient for this system to be based on a simple honest majority assumption.

Users are incentivised to participate in this monitoring market by the promise of a reward for being correct, with the optional introduction of additional incentives for staking early in the process if required. This is funded by Nahmii's 'Data Availability Bond'. The bond is accrued from Nahmii transaction fees and a portion is available as a reward for staking correctly. By only awarding a fraction of the bond as a reward for identifying when data availability changes, we ensure that there is always a reward for reversing any status change.

The specifics of the Data Availability Bond, such as currency type, maximum value and funding percentage accrual from Nahmii's fees, will be governed by the Nahmii Foundation. Token holders may also have a part to play in this governance, at the discretion of the Foundation. In addition, it is likely that this bond should be periodically converted to a mix of currencies such as ETH and stablecoins. It should not be majority denominated in NII, because of the relationship between Operator behaviour and the NII token price. If the Operator was performing data availability attacks, the value of the NII token and thus the bond would likely decrease substantially, creating a negative feedback loop for security.

The basic structure of the Oracle staking process is easy to understand: participants stake their NII tokens on true or false for a particular statement. Once the relevant criteria for that staking market are met, the market is considered resolved in one direction or another. Tokens of users who staked on the losing side are seized and distributed to the winners as a reward. The key concept here is that participants must stake something of real value and that the penalty for being a bad actor is maximally severe, i.e. that the user loses their staked tokens. This system of rewards and incentives underpins the core game-theoretic design principles of the Oracle. Put simply, we utilise the power of a distributed consensus-based market to trustlessly bring a critical decision process to Nahmii.

The Oracle will function entirely on chain as a true decentralised process, with no possibility of centralised interference. Therefore, the Oracle must be optimised over time and is still in testing. It must be able to quickly and accurately resolve whether data is or was available, yet be Sybil and, ideally, 51% attack resistant. This is non-trivial, but we have performed research and public testing which demonstrates that this is possible. This testing is described in the 'The Lottery Oracle Test' section below.

As part of the foundation model, discussed earlier in this paper, Nahmii Foundation members will be responsible for ensuring a plurality of token holders, minimising the risk of attack on the Oracle. Similarly, key Foundation members will be required to host mirrors of Nahmii's transaction data. This will help mitigate some data availability false alarms.

Data Availability Committee

Whilst the Oracle is in the testing phase, Foundation members (excluding Nahmii AS) will act as a Data Availability Committee. They will be granted the ability to 'vote' to pause a settlement if they become aware of missing data. This results in Nahmii being immediately decentralised and protected by reputable third parties.

Fisherman Attack

There is a risk that a compromised Operator could withhold data temporarily, so as to manipulate users into staking tokens in a particular direction (i.e. data was not available), only to later reveal the data at the last moment. In a naively designed system this would allow the Operator to effectively steal the tokens of users who had staked tokens in good faith, either by working with a third party or through staking tokens directly on the market in the opposite direction (which is now correct). This is known as the 'Fisherman Attack'.

In the example test statements provided below, we suggest a simple but effective mitigation against the Fisherman Attack issue. By including a test statement which references this type of Operator behaviour explicitly. It ensures that even if the Operator reveals data later, the answer to the test statement remains unchanged.

The Lottery Oracle Test

In 2019 Nahmii AS ran a public test of an Oracle that was designed to answer a lottery-related test statement:

'The 'bonus ball' drawn in the last UK 'lotto' draw was between 1 and 20 (inclusive)'

This test demonstrated that a relatively simple set of rules could encourage users to monitor a real world situation and coordinate their efforts to extract a small potential reward. Most importantly, users were strongly encouraged to act in good faith, as the risk of lying was high.

Full details about this public test can be found at the following links (note that Nahmii AS was formerly named hubii AS):

- Official announcement (Medium)
- Smart contracts and technical information (Github)

It is important to note that the Lottery Oracle example was not vulnerable to a Fisherman Attack as described above, because Nahmii AS did not control the results or the publishing of the UK 'lotto' data. As such, the Operator could not withhold data only to reveal it later.

A new set of public tests will be performed during 2021, to demonstrate the latest research on the Oracle.

Suggested Oracle Test Statements

It is recommended that the test statement includes a temporal element. This allows data availability validators to delay their staking slightly if there was a temporary data availability issue. This potentially allows the Operator a few hours to resolve the issue. Not only will this help avoid unnecessary false alarms, it would save validators from spending what might be considered unnecessary gas on a temporary issue.

A simple to understand test statement is:

"There has been a missing piece of data from an ongoing settlement for x hours or more"

The test statement set out above is not necessarily the final staking market that we will use; however, it serves as a useful indicator as to where the Oracle is heading. This statement allows x hours for the Operator to rectify temporary data availability issues.

Optimisation of the Data Availability Oracle

Further refinements to the Oracle's design include each settlement being monitored by a dedicated Oracle instance. This allows branches of Nahmii to be settled in the event of some data availability issues elsewhere in the network.. Ostensibly the system is operating as normal for wallets which were unaffected by any potential security issue elsewhere. The advantages of such a solution could be offset by the potential on-chain gas requirements, which would increase rapidly in the event of a

mass exit or global Operator fraud scenario. Similarly, this would raise questions around potential Sybil attacks and further dependencies on Ethereum.

The mitigation against these risks is to set a limit for how many settlements could be flagged as having missing data, beyond which all settlements would be paused until the situation was resolved. A rogue Operator could try to attack the protocol by starting a large number of settlements with missing data in order to trigger global settlement pausing, but this risk can be effectively mitigated by requiring the relevant number of settlements to be high. This would ensure a large amount of capital was needed to begin many settlements with associated Settlement Bonds, increasing the potential cost to the Operator. There would also be a significant capital cost in tying up that capital for an extended period of time. Lastly, given that Operators are required to post a large bond in order to participate in the running of Nahmii, they are also financially bonded to the success of the network. As such, attacks of this nature are unlikely to occur.

Active staking as discussed in the 'Active Staking' section also helps to reduce gas cost and burden on layer 1 in the event of any wide-scale data availability issues.

Advantages of a Data Availability Oracle vs. Alternative Consensus Mechanisms

There are a multitude of consensus mechanisms that could theoretically be used or adapted to answer a question of data availability for Nahmii. Data availability issues have a relatively unknown frequency, which might change over time as Nahmii is optimised. It is also the case that the Operator can be innocent or guilty. The Operator will naturally be punished as a Nahmii token holder because the price of the token will reflect usage, where usage will drop if the system does not work optimally. However, there must be an independent body that actually keeps the Operator honest, ultimately preventing any settlement with missing data being completed.

A settlement with missing data has a settlement period which must be long enough for consensus to be achieved around whether data is missing, but also to revert if that data was only temporarily unavailable. A change of data availability status for a settlement should restart or extend the initial Challenge Period, so there is always time to achieve consensus again.

The Oracle is built solely on Ethereum and therefore inherits both the performance issues and security of the platform. An Oracle is a very efficient way to achieve a consensus on Ethereum itself, as the Oracle is only required when something has gone wrong. The Oracle therefore functions on an 'approval by exception' basis, leveraging the likely infrequent nature of issues with Nahmii. This contrasts with many other consensus methods, where continuous and frequent Ethereum transactions would be required. Importantly, these alternative consensus methods often limit rival solutions in terms of transaction latency and finality - neither of which are a problem with Nahmii.

A well-designed Oracle can be a very secure and robust method to achieve 'slow' consensus over a Challenge Period. The blockchain industry has a general focus on achieving consensus as fast as possible whilst retaining security, but this is not strictly necessary for Nahmii. Given that users can exit Nahmii via liquidity providers without going through the settlement process, we are more concerned with security than speed. These alternative, but safe, exit options dictate that only advanced users will interact with Nahmii's settlement contracts. As there is reduced concern about the user experience around settlements, this allows us to ensure that the Challenge Period is a sufficiently long time period for an interactive game such as that proposed in the Data Availability Oracle.

For an infrequent 'slow' consensus, an Oracle is an ideal solution and can achieve security levels comparable with that of Ethereum.

Active Staking

Nahmii will require the active staking of NII tokens in order for those tokens to be eligible to earn transaction fees from the network. We must always be conscious of placing additional transactional load on layer 1, but active staking maximises flexibility around the data availability mechanisms of Nahmii. There are two primary reasons why active staking is important.

Efficiency of the Data Availability Oracle

It is key that to protect Nahmii from data availability attacks, there must be active engagement in the Data Availability Oracle when required. The prior deposit of NII tokens into a smart contract, ready for staking into the Data Availability Oracle is more gas efficient on layer 1. This reduces the cost of initial staking for any token holder and helps reduce the possibility of Nash equilibrium, where the optimum strategy for token holders might be to wait for other token holders to stake first.[6]

Verifier's Dilemma

The Verifier's Dilemma is a theoretical construct where an incentive based security mechanism is so well designed that no attacker ever attempts an attack, or attacks are very infrequent. Naturally the verifiers, who are expending resources to do their work, might stop watching. If they do, then this opens up the opportunity for an attack to take place.

In practice, for Nahmii, there are many reasons why verifiers would continue to monitor the network even if there was no direct incentive. For example, if you are the owner of a product which is built on and relies on Nahmii, then you are incentivised to secure the network. In addition, NII token holders have a direct stake in the network; they have value at risk if Nahmii's security fails. We can observe

[6] Nash equilibrium

this kind of indirect incentivisation for numerous blockchains, where people still run nodes even without direct incentivisation.

Whilst we do not view the Verifier's Dilemma as a major issue for Nahmii, we do think it is shortsighted to suggest it cannot become a problem. Active staking is a tool that can assist, as it allows the network to seed Data Availability Oracle instances with small but frequent direct incentives whilst minimising the gas required used for staking on the layer 1.[7] It is also possible to implement slashing for tokens which do not take part in any data availability staking that does occur. This can even help simplify the Data Availability Oracle.

Transaction Ordering

The finality of transactions on the Nahmii protocol is determined by publication of data; when the appropropriate receipt is published, a transaction is considered final. We also utilise the concept of checkpointing; each time a new transaction is confirmed through the settlement process, all transactions in its history are considered checkpointed. The full ancestry of a transaction may be a complex branching structure and contain many thousands of previous transactions across different wallets. With checkpointing, validators and users only need to be concerned about the recent ancestry dating back to any previous checkpoints in the family tree.

In the event that Operator fraud is detected, users will be able to settle their account up to the 'last known good' transaction. Any transactions dependent upon a fraudulent transaction are tainted and must never be settled; to do so would risk compounding the prior fraud.

Nahmii will use either Merkle trees or accumulators to ensure sufficient information is encoded within a transaction receipt, such that a transaction's ancestry can be fully reconstructed trivially and checked by validators. Proofs of inclusion and proofs of exclusion can be used to ensure that no transactions with a fraud in their ancestry can be settled. We are currently testing both Merkle trees and accumulators to determine which approach is optimal; there are differences in computation for construction and validation which should be optimised for latency, but we must also consider the proofs and the associated gas usage for on-chain submission of those proofs.

Operator Bond

The Operator Bond is a key protection mechanism against Operator fraud. The Operator (or multiple Operators) must lock up a large bond of some quantity of NII and other currencies into an Ethereum smart contract as a condition of being able to process transactions using Nahmii. This bond can be slashed to punish the Operator in case of proven fraud. As discussed earlier for the Data Availability

[7] Such small and frequent incentives could be trivially redirected from the transaction fees generated in Nahmii

Bond, the denomination of this Operator Bond in NII alone would be insufficient to protect the network.

If the network is attacked by a compromised Operator, the price of NII should drop. This is one punishment for the Operator, as their bond is no longer worth as much as it was previously. Unfortunately, the network would now be protected by a bond which is reduced in value creating a potentially dangerous feedback loop. Given that the cost of any validators submitting a fraud proof is paid in ETH, it is possible that the reward no longer covers the cost of identifying Operator fraud if the reward is solely in NII.

As such, the Operator Bond should also consist of currencies other than NII, preferably a combination of ETH and perhaps some stable coins. This bond can be part-funded by the transaction fees accrued by the Operator's NII. These fees will be accrued in a basket of many currencies, but some logic can be added to convert these currencies to ETH or stable coins via a suitable decentralised exchange. Additional logic will ensure this bond is always topped up sufficiently, whilst allowing the Operator to also withdraw some of their earned fees. If the bond is diminished due to fraud claims, then the Operator should not be able to claim their fees until the bond has been refreshed. The logic behind the bond and the size of it should be under the governance of the Foundation, who may also choose to include NII token holders in the decision-making process.

The NII locked up in the Operator Bond cannot be used for staking in the Data Availability Oracle, which ensures that the Operator has reduced power in defending against problems that they may have caused.

Specific Security Considerations

Transaction Acceptance

As Nahmii works in a similar fashion to a state channel, albeit with pooled security, there is some 'personal' responsibility for transaction acceptance. A user must only accept a transaction as valid

if they have checked the ancestry of that transaction in full. This means checking that all relevant data to their transaction ancestry is available and, crucially, that it is valid. Importantly, users should only be required to check non-checkpointed transactions; any transaction that was included in a previous successful settlement can be considered safe.

We do not anticipate that individual users will check their transaction ancestry, although the option will be available to them at all times. We anticipate this task to be performed in the background by the user's client software (e.g. their wallet). It is also possible to rely on third-party services such as an explorer, though this is less desirable.

The best-practice behaviour of a Nahmii client should be to ensure a local copy of all data pertinent to a transaction has been downloaded prior to accepting a transaction as valid. If there is any invalid or missing data in the ancestry then a transaction would not be accepted; any transactions which took place after that troublesome point are considered to have not taken place. In the case that there was a temporary disruption in data availability, the network is effectively brought up to date once the relevant data has been published. As mentioned, light clients which don't download all data, but use third-party services are also possible.

Efficient Settlements via Checkpointing

The settlement process for Nahmii allows for the optimisation of a user's settlement requirements. When a settlement takes place, validation is actually being performed for all transactions in the ancestry of the one being settled. Upon successful settlement, this means that any transaction in that ancestry was validated in terms of correctness and also availability of data. We might say that such transactions have been partly settled already, although those transactions will not have been checked against state changes that may have happened later.

For an individual user with one of these part-settled transactions, it is still possible for them to have performed later transactions that would be in conflict with their part-settled state. Nevertheless, the part-settled status of those transactions permits an abbreviated settlement process. The user merely needs to provide a simple inclusion proof to demonstrate that their transaction was partially settled and begin a Challenge Period. The user will still need to provide a Settlement Bond, to reward any challenge. The challenge itself is lightweight and does not require consensus about data availability and as a result we can reduce the Challenge Period in such circumstances. We anticipate that a significant proportion of settlements will be able to take advantage of this facility.

The settlement mechanics around part-settled transactions allow for an additional possibility of a market for widely connected and efficient settlements. It is possible for users to pay for a service

where an entity will 'connect' transactions together and regularly bring many wallets up to the partly-settled state.[8]

Similarly, Nahmii supports the concept of a direct settlement market, which enables users to exit the protocol without waiting for a settlement to complete. In this scenario, one user would provide on-chain liquidity to another in exchange for the proceeds from their ongoing settlement. The liquidity provider would be responsible for checking a settlement's validity before taking it over, to de-risk their position. This is an example of how Nahmii can provide near-instant settlements and withdrawals.

Challenge Period

As with many layer 2s there must be a Challenge Period where any malfeasance can be challenged, according to the specific security model. For Nahmii, a fraud proof can be provided at any time, by any user. The key reasons for a Challenge Period on Nahmii are to prevent users settling on old states and to give sufficient time to reach consensus about whether there are any data availability issues related to an ongoing settlement. A fraudulent settlement attempt will always reveal either a fraudulent transaction or the fact that there is missing data in that transaction's ancestry.

The Challenge Period does **not** impact the instant finality of Nahmii. Any transaction can be verified and accepted by a user within Nahmii immediately. It is still important to have a Challenge Period to protect the network from fraudulent activity.

The Challenge Period should also be a long enough time window such that the Ethereum network itself could not suffer from a Denial of Service (DoS) attack, preventing fraud proofs or consensus about data availability being reached. It is anticipated that Nahmii 2.0 will begin with a Challenge Period of 5 days (120 hours), which is the same as Nahmii 1.0. This can be shortened or lengthened as required. It is also possible for this duration to be dynamic based on prevailing network conditions. We would suggest that the Challenge Period for efficient settlements discussed in the previous section could be reduced to 1 day (24 hours).

If the Challenge Period ends with no successful fraud proof provided and no concerns around missing data, the Challenge Period ends and a user's state is settled. If relevant, they can withdraw their funds.

We anticipate that the overwhelming majority of users will never perform a settlement on Nahmii. For example, there will be a number of mechanisms by which users will be able to move funds from Nahmii to Ethereum and vice versa, without having to go through the settlement process. In

[8] This could also be performed by the Operator, who could send one small-value transaction to many wallets before effectively settling them all with a single settlement.

addition, products can be built on our protocol where the user will never need to know that the underlying infrastructure is Nahmii. In these cases, market makers, arbitrageurs and product owners will be the primary users engaging in the settlement process. This is one potential reason why the Challenge Period could be lengthened, as most users will never use this. There is an economic balancing act between the cost of capital for those parties providing liquidity into the ecosystem and any additional security benefits from extending the Challenge Period. Any suggested changes of duration is a governance decision, which should be brought forward to the Foundation or for a vote by token holders.

Bonded Settlement

As Nahmii moves to support generalised computation, we will move to a system of bonded settlement. A bonded settlement means that a user wanting to settle their account must put up a bond to begin their settlement. The bond can be reclaimed upon successful settlement.

There should always be a minimum reward available in order to challenge a fraudulent settlement attempt. The challenge by a validator takes place on the Ethereum base layer and costs gas in the form of ETH. As such, the bond to begin a settlement should be denominated in ETH and be sufficient to cover the gas cost of a challenge. Preferably this should be a profitable exercise by a validator, to provide incentive to perform this task.

It is possible that the amount of the ETH-denominated bond to begin a settlement will seem high. As discussed previously, we expect the majority of settlements will be performed by market makers, arbitrageurs and product owners and not by regular users. Importantly, the cost of capital associated with locking up this ETH will be low due to the short duration of the settlement Challenge Period.

Mass Exits

A mass exit from a layer-2 protocol is where a large number of users attempt to settle their wallets and potentially withdraw funds at the same time. This can be a problem for systems which rely on a slower layer-1 blockchain to process settlements, especially if users have a limited time to exit and the network is congested due to the influx of new transactions.

Mass exits are less of a problem for Nahmii than other layer-2 protocols, mainly due to our efficient checkpointed settlements and the Data Availability Oracle. Given that one settlement can potentially checkpoint an infinite number of transactions, even a small number of successful settlements in a mass exit scenario would allow most users to withdraw without needing to settle their own wallets.

The key security concern for Nahmii when faced with a mass exit is the continued protection of the Data Availability Oracle. The Oracle relies on Ethereum to function, so it inherits any problems with Ethereum's availability. It is possible to run the Oracle as a test for data availability against individual settlements, rather than the general question of whether data is available overall. A mass exit could therefore generate a large number of new settlements which need to be monitored simultaneously by the Oracle, which could be operating with limited throughput due to Ethereum-related concerns. One potential solution to this issue is described previously in the 'Optimisation of the Data Availability Oracle' section, where we suggest employing both individual Oracle instances for settlements and a complementary global data availability protection.

Miner Extractable Value

An oft-discussed problem in the layer-2 space is the concept of Miner Extractable Value (MEV). This is related to the ability of miners to extract value from users on Ethereum, by ordering transactions in a manner which favours themselves or their partners. This is exemplified in particular for Automated Market Maker (AMM) Decentralised Exchanges (DEXs) where miners can order transactions to guarantee themselves arbitrage profits. Although we discuss miners here, being the terminology for Ethereum 1.0 PoW consensus, the change to Ethereum 2.0 PoS consensus does not alter the message in this section.

It is our belief that the solution to these problems lies not in the complex suggestions other layer-2 protocols make, such as to set up MEV auctions, nor to add in the complexity of sequencers. The best way to control such practices is to reduce the incentive to do these things, by increasing the liquidity and efficiency of a marketplace. This reduces the potential benefits and increases the cost of capital of taking advantage of any opportunity. In the case of scaling solutions which have one or a limited number of operators, you want the balance to be tipped towards them having more benefit by running the market properly. This is the case for a solution built on Nahmii, because any attempt to extract value in this manner will be easily identifiable. The Operator processes all transactions and publishes them off chain, placing full responsibility on themselves. If the Operator behaves badly, they will be held to account for their actions publicly; their reputation damaged.

It is our opinion that MEV on a layer 1 can help to create a more efficient market. In a typical AMM arbitrage opportunity that is coalesced and taken advantage of by a miner, the trading volume is effectively tripled. This can lead to increased trading fees and, consequently, larger pools of liquidity because of those increased fees. An increased pool size then reduces slippage for the equivalent-sized trades. In a sense, this has ensured greater utility for those AMMs on layer 1.

The Real Danger of MEV

We would suggest that the understanding of MEV within the blockchain community is incomplete. There is a widespread short-sightedness that MEV is limited to value extracted on chain by miners

reordering transactions. This is not the case; miners play in a world that is not restricted to what happens on the blockchain. The topic is too large to be completely fully here. Instead, we will illustrate a simple example and leave it to the reader to consider the wider consequences.

All other layer-2 solutions which rely on periodic commitments to a layer 1 have a critical vulnerability. Nahmii does not have the vulnerability described below.

To a miner, any layer 2 can be perceived as a parasitic threat to their revenue. During 2020 and 2021 gas prices reached levels which cause transaction fee revenue to be equal to the block reward for extended periods of time. We do not anticipate that this situation will change over the long term. Any solution which takes transactions onto another layer and increases the overall network capacity, has the potential to reduce congestion on layer 1, thereby reducing the total revenue for miners. It can be argued that there is so much pent-up demand for Ethereum transactions that the capacity will always be filled, but this is unclear.

The design of almost all other layer-2 solutions necessitates regular commitments in the form of transactions on Ethereum. This puts the success of those protocols firmly in the hands of those miners, who may feel aggrieved by any considerable level of layer-2 success.

All miners of a reasonable size have the ability to DoS any layer 2 which makes periodic commitments. If they have only 1% of the mining capacity of the Ethereum network, then their DoS represents a minor reduction in capacity on that layer 2. If a majority of mining power determines that a layer 2 is parasitic then they can DoS that layer 2 completely, or at least hold the protocol to ransom. They may extort a high transaction fee for the necessary commitments in order to compensate them for perceived lost revenue. This can be the rational thing for miners to do and would be analogous to the MEV discussed widely in the Ethereum community today.

Unfortunately, this is not the worst of it. In addition to simple extortion of high transaction fees from those layer 2s, miners can extract considerably more value through the manipulation of token prices. Many, but not all, layer 2s rely on a native token for governance or security of the network. It would be trivial for miners to short the value of those tokens and then DoS the layer 2.[9] A token that is intrinsically linked to the value of a network which is not functioning will most likely fall in value, with the miners profiting.

Even commitment-based layer 2s without native tokens remain vulnerable to a shorting attack by miners. Although these layer 2s don't have a native token, this is not necessarily true for the projects built on top of them. In some ways this is worse, as these projects are vulnerable to shorting attacks through no fault of their own. Layer-2 project tokens can be shorted by miners whilst they DoS the

[9] Short (finance)

underlying layer-2 protocol. Miners will still profit from such attacks, to the ultimate detriment of the layer 2 once again. Overcoming such an attack can only be done by increasing the gas price for protocol commitments or by attempting to directly tip miners to include the commitments on the layer 1. The cost of doing so under a sustained attack can rapidly become unaffordable for users, projects or the Operator of the protocol.

Nahmii was designed from the ground up so that it does not have this vulnerability. Whereas other layer 2s need regular commitments, Nahmii only needs infrequent transactions on Ethereum (such as when settlements take place). Additionally, a single settlement can be used to part-settle countless wallets or transactions at once, lessening the load on the base layer even more. Even if such transactions are expensive due to miner extortion, this infrequent cost can be absorbed more easily. Significantly, this will not affect the transactional performance of Nahmii; it only affects settlements, primarily fund transfers between layer 1 and layer 2. This will, in most cases, only increase costs for the likes of market makers and arbitrageurs, but we would not anticipate this to materially impact the efficiency of the market between layer 1 and layer 2. There is also the ability to fine tune settlement Challenge Periods to increase resistance to any DoS based attack, such that this problem will never occur for Nahmii.

Sword of Damocles

There is the potential for an Operator to refuse to publish either a transaction receipt or acknowledge transaction failure in respect of a user's transaction request. In such a scenario the user does not know if the transaction was completed or not and cannot begin a settlement on their latest transaction as they lack the appropriate evidence to do so.

This is solved using a simple interactive game on layer 1. Any user who has been left in a precarious state of limbo has a guarantee that they are able to use this mechanism to extricate themselves from the situation. It is unlikely that this situation will arise because of the availability of this mechanism coupled to the Operator's financial bond to Nahmii.

The process, which can be automated by software, would involve a user, Alice, creating a new and valid transaction request for layer 2, but submitting it to a smart contract on layer 1. This transaction request would have the same nonce as the transaction being withheld by the Operator, but would be one which sends Nahmii's native currency from the wallet in question to itself.[10] This is essentially a request to cancel the transaction in question. At this point the Operator (or anyone else monitoring the network) must either publish the countersigned original transaction or the new cancellation transaction on layer 1, within a set amount of time. If the Operator continues to withhold either of these transactions past the deadline then the Operator can be punished via slashing their Operator Bond.

[10] A nonce is an incrementing integer which denotes the transactional order of transactions made by a wallet.

If the Operator supplied a transaction receipt then Alice will be allowed to settle safely, using whichever receipt was provided by the Operator. If the Operator did not provide a transaction receipt and was punished, then Alice will now be allowed to begin a settlement on the transaction prior to the one that was in dispute. No later transaction will be allowed to be used to challenge this settlement.

It is important that any other user can challenge Alice's cancellation request in order to prevent collusion with the Operator. There is no proof that Alice is a guilty party though, and so the reward for any other user challenging this cancellation will be paid from the Operator Bond. This reward must cover the cost of the challenge, which will cost the user gas on layer 1.

If the transaction was withheld or ignored by the Operator and was never seen publicly, then it is treated like it never happened. No user would have accepted a transaction that was never published, nor would they have accepted a transaction that was built upon such missing data.

Schrödinger's Sword of Damocles

A consideration must be made for a scaling solution which is hugely scalable, but relies on data availability. A user (or their software) will treat a transaction as final once they receive a receipt from the Operator, provided that they have also validated the transaction ancestry back to the previous checkpoint (if applicable) and ensured that all data was available. This is fine when all data is available, but Nahmii must also have an answer for when there are some receipts missing in the transaction ancestry. At this point the user doesn't know if the transaction really took place or not. There are two possible situations here:

1. The Operator has withheld receipts which would prove fraud
2. The Operator has accidentally withheld receipts which are valid

If the Operator reveals the data and those transactions are valid, then the user's later transactions have effectively taken place. If the Operator reveals the data which proves that transactions are fraudulent, then those later transactions are considered invalid as they are built upon fraud. The Operator will then be punished via the regular fraud prevention challenges and can expect to lose some of their Operator Bond.

If the Operator *never* reveals the transaction data, users could be left in limbo as to whether their transactions have taken place. Here is where we can add a relatively simple interactive game where any user can ask the Operator to publish those missing receipts within a reasonable length time window. If the Operator refuses to publish receipts then those transactions must be considered fraudulent and the Operator punished.

The exact implementation depends on how Nahmii is built, but should ensure there are no edge cases in this interactive game where the Operator can profit from a Fisherman Attack by revealing data later or, more importantly, where an attacker can force the Operator to spend excessive gas defending themselves. Such an attack against the Operator is not free though, as the interactive game would cost an attacker gas on Ethereum also. We may also choose to require the user to pay a small fee to the Operator for this service, which should cover the Operator's costs and mitigate against this kind of attack.

For any user caught up in this situation, they can always exit the system so long as they have all their own data and can see that their later transactions do not 'overrun' their state. For a user whose transactions are part of the missing data, then they should engage in the interactive game or wait for someone else to do so, so they can be certain of their own situation. The situation may take a short while, but will ultimately be resolved correctly.

Imagine an implementation of a Uniswap-like trading system on Nahmii. As Nahmii is designed to be highly scalable there may be hundreds or many thousands of transactions per second on this smart contract. All of these transactions are built upon one another and so, if an arbitrageur or market maker is left in limbo with missing data, their trading could be affected. This would present similar difficulties to trading on a centralised exchange where their connection is lost or the exchange goes offline. It is problematic, but is part of the risk that those individuals take. The key takeaway here is that the situation can always be resolved and that there is a strong incentive for the Operator to run the system as smoothly as possible. This will ensure a great user experience and filter down into benefits for the Operator themselves, in the form of higher transaction fees and increased confidence by arbitrageurs which improve liquidity across products.

'Alternate Reality' Issue

The 'Alternate Reality' issue refers to a problematic scenario in which two or more transactions on the layer 2 have been assigned the same wallet nonce. We know that at least one of these transactions must be fraudulent and that the Operator has been compromised, as two transactions sent by the same wallet cannot share the same nonce. When presented with these transactions, both of which have been signed by the sender and Operator, how can we determine which transaction is valid?

Each transaction within Nahmii results in an incremented wallet nonce. As with Ethereum, Alice signs the transaction with her suggested transaction nonce. In the case of a simple payment between Alice and Bob, Alice's payment should increment her wallet nonce.

The ‘Alternate Reality’ issue arises when a compromised Operator accepts the same wallet nonce for multiple transactions. Alice may be innocent, because her software may have made the error, but she may also be colluding with the Operator. As we cannot know for certain whether Alice’s actions are deliberate, the blame for this should always lie with the Operator who we expect will never countersign two transactions in this way. A simple example is included below:

1. Alice’s balance is 100 tokens
2. Alice signs a transaction to transfer 50 tokens to Bob
3. Alice signs another transaction with the same wallet nonce to transfer 50 tokens to Carol
4. The Operator countersigns both transactions

Both transactions are valid in isolation, but cannot exist together. There are now three possibilities:

a) The Operator publishes both transactions
b) The Operator publishes only one of the two transactions
c) The Operator does not publish either transaction

In the first case, we have a clear case of Operator fraud as there are two transaction receipts showing the same wallet nonce. These receipts can be presented to Nahmii’s underlying smart contracts, which will punish the Operator. This is the correct outcome; however, there is a problem. Viewed from the perspective of the smart contract, it cannot determine which of the two transactions is valid. Both have been signed by Alice and the Operator, but we cannot say exactly when they were signed or published. All we know at this stage is that both transactions cannot be valid at the same time.

If only one receipt is published, there is a slightly different problem. If Bob’s receipt was published, Bob would expect his transaction to be final. Using this receipt, Bob could now begin the settlement process to return these tokens to the mainnet. Assuming that Carol’s receipt is not published during this time, there is no evidence to suggest any fraudulent activity. Bob’s settlement should therefore go through without challenge. Presumably, this means that Carol’s unpublished transaction was the fraudulent one, given that the Operator never revealed it. Assuming Carol was innocent, she never received a receipt and therefore her transaction effectively never happened; she has no expectations of having received funds. If Carol’s transaction is subsequently revealed by the Operator, it is too late to negatively impact the innocent party, Bob. Carol would be correct to reject the transaction as she would be able to see that it is fraudulent, because of the existence of Bob’s receipt. Both Carol and Bob’s receipts could still be used to punish the Operator. It is unlikely this would happen though, given the Operator would effectively admit to wrongdoing and receive punishment with no possible benefit.

In the final scenario, neither transaction is published. Assuming both are innocent, neither Bob nor Carol should be aware of receipt of any funds. If Bob or Carol have copies of the receipt, but nobody else does, then they might try to spend these funds. In attempting to spend these funds, then they will at least reveal missing data and their attempt to spend funds should be rejected. This is exactly how we envision payments for goods using Nahmii to work; the merchant would use software to check the purchaser's transaction history. Such a check would detect the anomaly and reject the transaction. Neither Bob nor Carol would be able to settle and withdraw their funds from Nahmii, as they would have no relevant receipt to begin the settlement process with. If only one receipt is published at some point, then we revert to the case described above where that transaction is valid unless the other receipt is published also.

In summary, the problem posed by the 'Alternate Reality' issue is limited to the situation where both receipts are published. Clearly the Operator should be punished, but we need to know how we decide which transaction is valid. The solution comes in the form of Nahmii's governing Foundation, who will be trusted to resolve these cases fairly. Later on, the solution to this problem could be handed over to an Oracle similar to the Data Availability Oracle.

Validity Proofs vs Fraud Proofs

Validity proofs are a technique used to secure some other layer 2s, namely ZK-SNARKS[11] or ZK-STARKS[12] used in the various forms of ZK Rollups. Whereas a fraud proof is used to show something has gone wrong in the past, a validity proof ensures that a state transition is correct as it takes place. Ostensibly there are some benefits for using these cryptographic proofs, but we will describe here why they can create serious and irreconcilable issues for scalability.

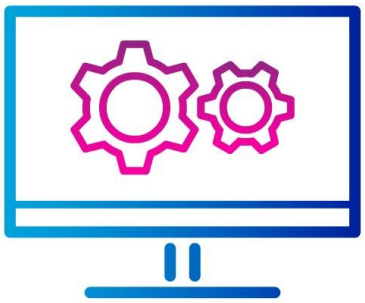
The Scalability of Validity Proofs

When it comes to the commercial scalability of validity proofs, there is no doubt that they create a serious bottleneck in transaction processing and consequently latency and finality. They are useful only for the increase of throughput.

A validity proof is computationally intensive to generate, but extremely easy to validate, even on layer 1. This means it is very simple to check if a transaction is valid when the proof has been committed to Ethereum. Whilst the generation of a validity proof for a small number of transactions can be relatively quick, it is impractical as there are only throughput benefits for batching large numbers of transactions together in the proof. The generation of a typically-sized validity proof for on-chain submission can take tens of minutes with current technology and implementations. For the foreseeable future, proof generation time will remain closer to minutes than the sub-second latency required for many commercial use cases.

[11] Zero Knowledge Succinct Non-interactive Argument of Knowledge.
[12] Zero Knowledge Scalable Transparent Argument of Knowledge.

This is additional latency over and above the latency native to any commitment-based scaling solution. Not only do proofs take a significant amount of time to generate, all users on a network must wait for those proofs to be finalised on Ethereum before they can be considered final. There are various claims of systems using sequencers or other methods of economic bonding to ensure quicker levels of finality, but these techniques add complexity, capital inefficiency and reduce security. Therefore we deem these methods to be impractical in general.



Nahmii Virtual Machine

Nahmii will add generalised smart contracts via a 'Nahmii Virtual Machine', NVM, which is designed to be compatible with the Ethereum Virtual Machine, EVM. Optimism is a project which has done an excellent job at porting the EVM over to a layer-2 protocol.[13] Unfortunately, the Optimism project does not exhibit many of the requirements for commercial scalability.

Essentially, the NVM will be an adaptation of the Optimism Virtual Machine, OVM, but suited to the specific architecture of Nahmii. This ensures a number of key characteristics for Nahmii smart contracts:

- Maintain the scalability of Nahmii
- Add the ability for generalised smart contracts
- Highly compatible with the EVM and Ethereum tooling, allowing smart contracts to be ported to Nahmii with ease
- Potential compatibility with Optimism and other EVM-based scaling solutions, which will help build a more powerful ecosystem
- Composability between smart contracts that live on Nahmii
- Familiar usage for existing Ethereum users, while also being much easier for new users vs. Ethereum

[13] Optimism project

Key Characteristics of Nahmii

Performance Summary

High Throughput

The nature of Nahmii's architecture provides horizontal and elastic scaling. Across the network as a whole there is **no limit** to the transactions per second that can be processed.

There will be natural bottlenecks at specific parts of the system due to the inherited limits of the software packages we use (such as I/O limitations of LevelDB in initial implementations). These can arise when there are abundant transactions that are connected to each other, for example by particularly high volume smart contracts for trading. The performance of these specific parts of the network will be similar to that found in 'real-world' trading engines. There is no diminished performance due to an Ethereum-related overhead.

Low Latency

Nahmii's architecture provides exemplary low latency (milliseconds) for transactions. A user's internet connection will usually be the limiting factor in processing transactions and giving feedback.

Instant Finality

As a consequence of Nahmii's state channel-like architecture, which avoids the need to commit regularly to the base layer, there is instant finality for all transactions. Once a transaction receipt has been received, it is irreversible.

Predictable Transaction Fees

Being a key feature to obtain wide commercial adoption, fees within Nahmii are always known prior to any transaction taking place.

Low Transaction Fees

The fees are set by the Operator and will always be competitive according to market conditions. Nahmii does not have a high operational cost, it is significantly more cost efficient than common tech stacks.

Fast Withdrawals

As described in this document, we view direct settlement on Nahmii as an expert process which will be mainly used by market makers, arbitrageurs and product owners. There will be multiple alternative methods for users to cheaply and quickly move funds out of Nahmii and back to layer 1.

The simplest method would involve liquidity providers who are willing to accept funds after the settlement Challenge Period in exchange for funds on layer 1 *now*. There would likely be a small fee for this service which would be correlated to the capital cost of funds being unavailable during the Challenge Period. Significantly there would be little to no risk component to the fee as the liquidity provider can check the settlement and be sure that there is no chance of it being unsuccessful.

As Nahmii usage grows over time, we expect there will be a significant amount of deposits and settlements which can be matched against each other. The fee for such a service would depend on the asymmetry of fund flows between layers 1 and 2. Such a flow can be optimised in a direct trading market, where layer 1 funds are tradeable with layer 2 funds of the same currency.

It is also possible to have fast and functional non-fungible token (NFT) withdrawals from Nahmii, even in the extreme case where the NFT is unique. It is unclear of the exact user journey, until the market demands such a feature. One can imagine a particular implementation where a settlement being processed on layer 1 involving an NFT could be deposited into a smart contract in order to 'mint' a clone of that NFT. The NFT 'in-flight' between layer 1 and 2 is inactive for the duration of the Challenge Period. The cloned NFT could be used during that same time period, though perhaps with restricted functionality, e.g. transfers could be prevented. That cloned NFT could then be traded back for the original NFT at the end of the settlement Challenge Period once the original NFT was released. It would be possible to enable transfers of cloned NFTs also, but this would open up the possibility for trading and naturally any buyer of that cloned NFT would need to check the provenance of that cloned NFT, to be certain that the Challenge Period for the original version would pass without incident. Otherwise the user might be left out of pocket. This is akin to the concept of wrapped tokens, but here the lifetime of the wrapped token is limited.

Transaction Fees

The transaction fee structure in Nahmii 2.0 will be significantly different to that in Nahmii 1.0. In order to charge fees for generalised execution of smart contracts we will leverage the method by which fees are charged on Ethereum. This means that fees are approximately based on the complexity of the code executed in a smart contract call.

As Nahmii will have an NVM, based upon the EVM, we will begin by using the same gas requirements for each transaction. As stated earlier, one of our key requirements for Nahmii is to have predictable costs, but how can we ensure this if we replicate the gas based model of Ethereum?

Defining Transaction Price Stability

Predictable costs to us means that transaction fees must be stable over extended periods of time. It is also important that transaction fees can be tweaked over longer periods of time to maintain business sustainability. There will be a variety of parameters that can be used to adjust fees, but suggestions by Nahmii AS must be brought before and accepted by the governance of Nahmii. As an example, over time we must tweak the gas requirements for different instructions in the NVM; there is a real world difference between the cost of these instructions being executed on Ethereum and on Nahmii.

Gas Price

The first thing we must remove is the variability due to the concept of a gas price market. Ethereum has a limited capacity, so a gas price was implemented, such that even with fixed execution gas there was still the ability for users to outbid other users and get their transactions processed more quickly by miners. As Nahmii has no such serious throughput limitations, we will implement a fixed gas price, thereby removing this congestion charge.

Fee Currency

On Ethereum, you must use ETH as a fee currency for paying for the gas consumed by contract execution. When it comes to commercial adoption there are two issues with this method. Firstly, a user or product owner must purchase ETH in order to use the Ethereum blockchain. This is the first barrier to wider adoption. Secondly, the price of Ethereum itself is volatile and so using ETH also introduces some fee variability.

This will be rectified in Nahmii by moving to a model where gas costs are fixed in USD terms; USD is still the world's reserve currency. The simplest, and initial, implementation is to use ETH as the native Nahmii fee currency, even if the fee is stable in respect of USD.

It is essential for a smooth user experience that other fee currencies are made available on Nahmii later. Imagine a user is transferring some tokens, the fee would ideally be in the same token as that being transferred. To enable fees to be paid in a range of tokens, exchange rate data is essential in order to maintain a fixed cost in USD terms. Initially, this can be achieved by a regularly updated pricing oracle, maintained by the Operator. Later on, this can be replaced with other suitable decentralised oracle implementations, as have been seen developed on Ethereum.

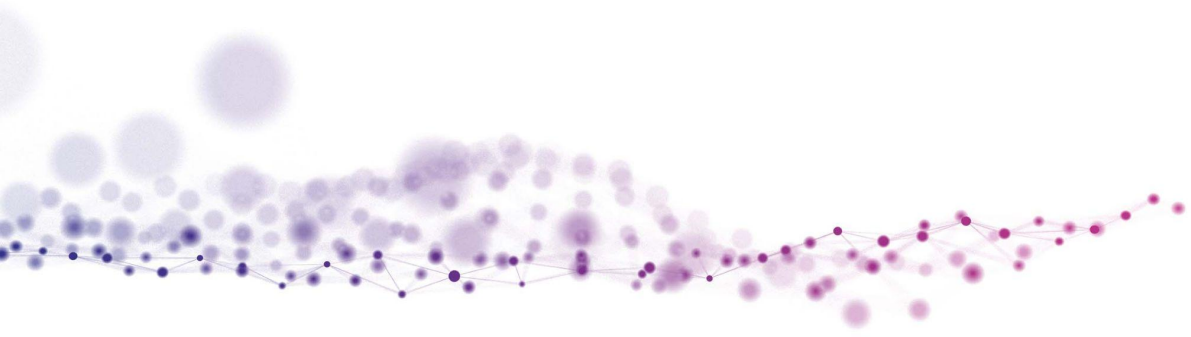
In Nahmii there will be many smart contract transactions where there is no natural or obvious fee currency, e.g. to pay for the transfer of a non-fungible token (NFT). In this case users would have to select a fee currency to be used to pay for their transaction. We may only allow a limited range of tokens for such transactions, though this is not essential.

There is no major technical challenge in the implementation of multiple fee currencies, but it is important to give users a satisfactory and non-complex experience. Fortunately, we have seen examples of multi-currency wallets in the regular financial world, where payments across currencies have been made seamless to users. The same principles can be applied for fee currencies.

Patent

Certain elements of the Nahmii protocol are patent pending. The decision to seek patent protection is driven by the need to keep the Nahmii protocol both open and democratic on a perpetual basis. There are two principles behind this decision: in the first instance, that the patent application will ensure that no vested interest can interfere with or prevent the protocol from being deployed and used; second, by vesting the patent in the hands of the Foundation we will demonstrate our faith in the Nahmii community to manage its availability and build upon it for the future. In the eventual situation where Nahmii moves to a DAO governance model, there will still need to be a legal structure for management of the patent.

As stated, when granted, this patent will be given over to the Foundation in perpetuity, conditional on the Foundation adhering to certain key principles. It will then be up to the members of the Foundation or token holders to determine a strategy for the Nahmii patent. It may be decided that the best strategy is to give the patent away; however, this does not undermine the logic of applying for the patent in the first instance as to do so gives flexibility around how to proceed.



Carbon Impact of Nahmii

The impact of Nahmii on carbon emissions will either be zero or create a net reduction in global carbon emissions.

There is widespread knowledge that Proof-of-Work (PoW) blockchains are extremely energy intensive. Many of the leading blockchains consume the same amount of energy as small countries. There is much debate about what fraction of this energy consumption causes 'additional' carbon emissions around the globe because nobody is truly sure what the electricity generation mix for the networks are. It is also true that PoW can be useful in load balancing electrical networks, which can have either a net positive or negative effect on emissions for a given consumption, depending on the local infrastructure for mining operations. There will eventually be a reckoning for these networks, based on the wider public good, the financial infrastructure that they displace and the net emissions they create.

Ethereum itself will undergo significant changes in the coming years, as it transitions from PoW to a Proof-of-Stake (PoS) consensus mechanism. This complex form of voting, by parties who are staking value into the network, will ensure security is maintained whilst also significantly reducing energy usage by at least one order of magnitude. This energy usage reduction will complement and further reduce Nahmii's energy usage.

The operation of Nahmii is comparable to that of a traditional technology business. Naturally, there is electricity consumed by any Operator on the network, by any user's device and by the infrastructure and products building on Nahmii. This electricity usage is more similar to that of a regular financial infrastructure business, but predominantly focused on electrical equipment such as servers; overall Nahmii would displace some need for ancillary services such as administration, auditing and customer service. The carbon impact of this electricity usage can be offset by the

appropriate parties for minimal cost. Overall any transaction that takes place on Nahmii will likely reduce carbon emissions compared with a regular technology business.

When it comes to carbon emissions there are three types of transactions on Nahmii :

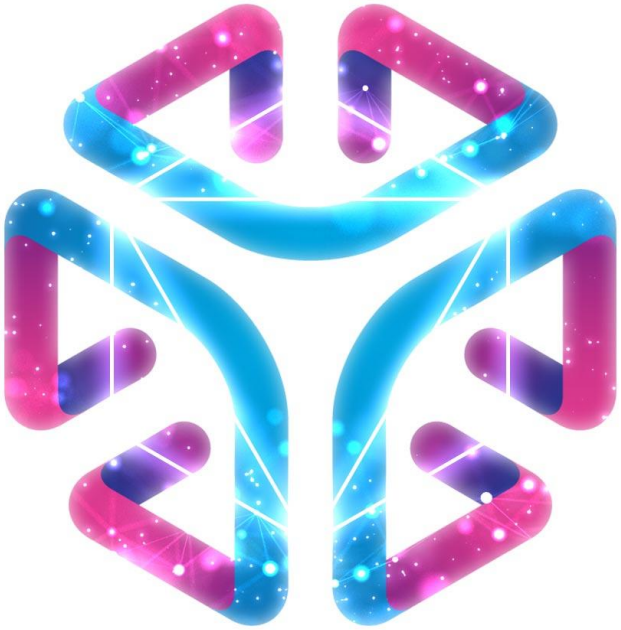
- First, there are the transactions that have been moved from Ethereum to Nahmii. Every transaction on Ethereum currently creates a number of kilograms of carbon emissions, our internal estimates being approximately 6kg per 21,000 gas Ethereum transfer. A transaction moved on to Nahmii causes a significant net saving of carbon emissions, as a Nahmii transaction creates negligible carbon emissions even if it is not offset.

- Second, there are transactions that are created by the emergence of Nahmii. These transactions are those that never would have existed on Ethereum without Nahmii. This can create net new carbon emissions, but they are minimal and can easily be offset. In almost all cases, they would be replacing transactions that took place in a non-blockchain system with significant carbon overhead. We argue that they will still represent net carbon savings.

- Third, there are the 'administrative' transactions in the operation of Nahmii. These take many forms, such as fraud proofs, settlements and deposits. The carbon impact of these transactions will be vastly outweighed by the carbon savings of transactions on Nahmii.

As Ethereum itself moves to PoS, its energy consumption and carbon footprint will be vastly reduced. At this point transactions moved to Nahmii from Ethereum will no longer represent such massive carbon savings, but equally Nahmii's 'administrative' transaction carbon load will be reduced. Overall we believe that Nahmii will reduce carbon emissions still when all sources of transactions are taken into account.

We will work with third parties to report the carbon emissions for all transactions on Nahmii and the potential carbon savings that they represent. It is highly likely that any usage of Nahmii will cause a net reduction in global carbon emissions, even after Ethereum moves to PoS.



The NII Token

Nahmii is a tokenised protocol. There are a total of 120 billion Nahmii (NII) tokens, which are held in time-locked contracts and released at the rate of 1 billion tokens per month over 10 years. The first emission was in December 2018.

All transactions within the Nahmii protocol incur a transaction fee, with fees accruing to token holders and Nahmii's community of protocol facilitators. The overwhelming majority of fees will go to NII token holders, with a small percentage being used to fund the Nahmii Foundation and Nahmii's Data Availability Oracle.

As tokens are released from the time-locked contract, they will be distributed accordingly:

- 50% proportionally to HBT token holders (if held in a non-blacklisted Ethereum address or Nahmii)[14]
- 40% to be used by Nahmii AS to develop the ecosystem, with oversight by the Foundation[15]
- 10% to the key partners that developed Nahmii

As stated, effectively all transaction fees will go to the token holders and facilitators of the protocol. It is essential that a significant percentage of Nahmii fees are shared with token holders. In a similar

[14] We reserve the right to blacklist certain Ethereum addresses from the HBT distribution, such as exchange addresses, with Foundation oversight

[15] Note, for the first 8 months 20% was airdropped to Ethereum holders that registered for the airdrop. More recently 10% has been used to incentivise liquidity provision on Uniswap.

fashion to many projects in this space, the security of the protocol is strongly related to the value of the token itself. As such, the token value acts as a bond for participants to ensure the correct operation of the security mechanisms. It is therefore critical to note that the funds that accrue to token holders is not a passive income; token holders must monitor, validate and secure the protocol. This is an incentive mechanism for participation, just as Bitcoin miners receive a mining subsidy and transaction fees.

Part of the remainder of the generated fees contribute to the 'Data Availability Bond' described in this paper. This bond, which increases over time, provides specific incentives for data availability validation and staking. This represents an additional reward for further active participation in the protocol for token holders, as only NII will be accepted for staking on data availability questions. Again, overall protocol value contributes directly to the difficulty of a 51% attack. As this value grows, a 51% attack becomes more expensive and the risk for an attacker increases.

Additionally there may be other minor security bonds. These bonds could be required to incentivise users to identify and punish Operator-only attacks which are identified later.

The exact division of Nahmii's transaction fees between token holders, Data Availability Bond and any other minor security bonds will be optimised over time. This long term optimisation will be a protocol governance decision.

It is suggested to begin with that transaction fees are split as per the following:

- 95% to NII token holders
- 2.5% accrued to the Data Availability Bond
- 2.5% accrued to the Foundation NII buyback as described in the 'Governance and Security Model' section

What is HBT?

HBT or Hubiits (Nahmii AS was formerly known as Hubii AS) is a token, originally designed in 2017. In 2018 we became aware that Plasma was not going to be commercially viable, which undermined our plans to use it for content micropayments (our pre-existing business). At this time, Nahmii AS began to build Nahmii to solve this problem ourselves. Nahmii AS later pivoted to a blockchain infrastructure company. As Nahmii AS did not require funding, we deemed that distribution via the HBT token airdrop was the most suitable method to reward our existing community.

HBT will continue to exist and we have products in development and production that can use HBT as a payment currency today. We do reserve the right to migrate the distribution of NII that currently goes to HBT holders to a new token if we deem it necessary (for example, a change of token name

or other token functionality upgrade). In all cases, the new token would be redeemable on a one-to-one basis with HBT to guarantee no material difference for HBT holders.

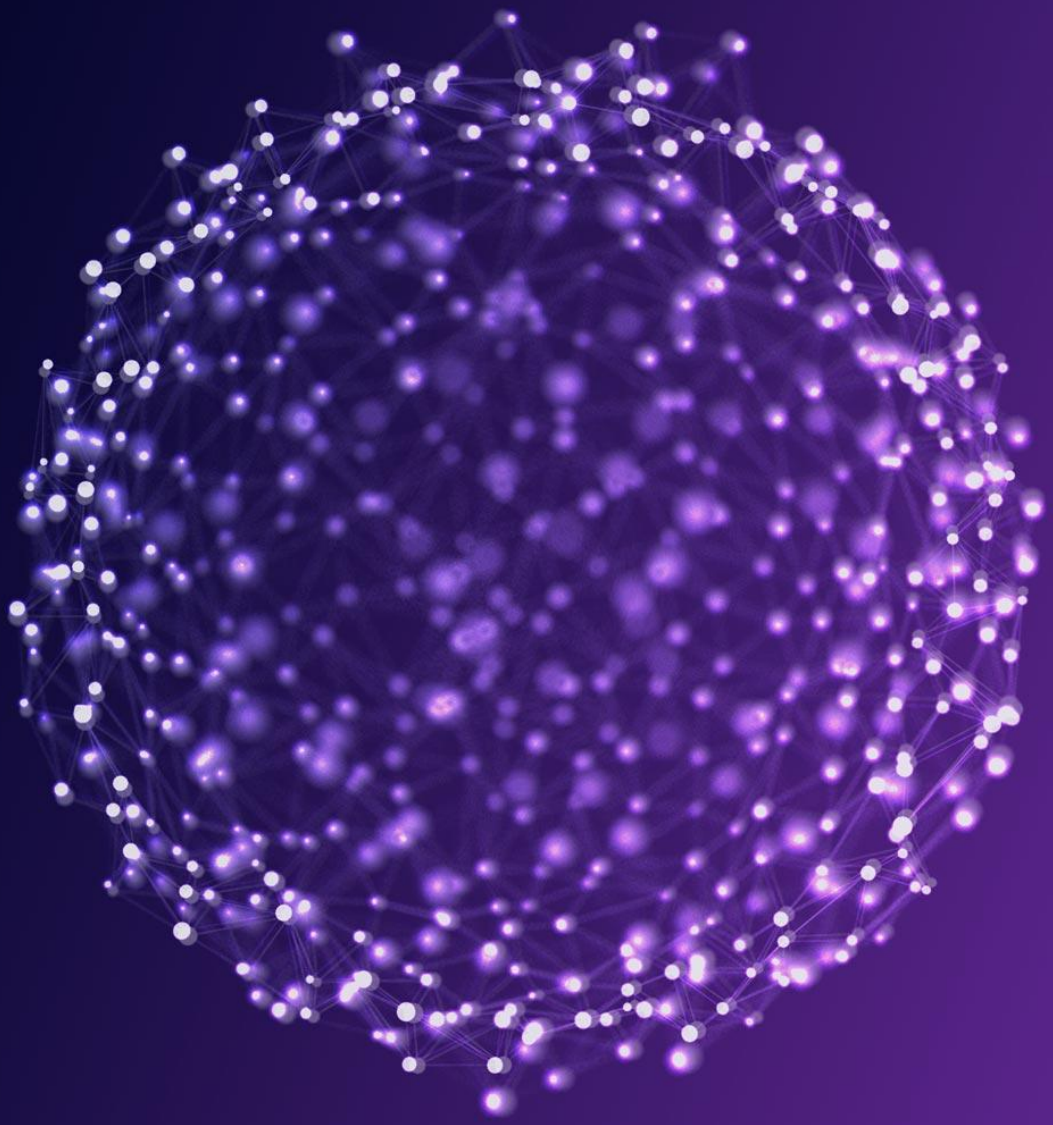
Balance-Blocks

In order to calculate the appropriate airdrop share for each address holding HBT (and ETH, whilst that portion was allocated), we utilised the concept of balance-blocks. Balance-blocks are designed to measure both the number of tokens held at an address and how that balance has changed over time, where balance is measured in tokens and time in blocks. More formally, the balance-block is defined as the integral under the balance versus block height chart for a given address across a specified period. One balance-block is therefore equivalent to holding one token for the period of one block.

The balance-block concept is sensitive to how a user's balance changes over time, ensuring that all token holders are treated fairly during the airdrop. This approach compares favourably with the traditional method of simply recording address balances at a fixed point in time and allocating tokens accordingly. In the traditional case, there is a strong incentive for a user to only hold HBT tokens around the time of the airdrop; a user who transfers 100 HBT into their wallet one day before the airdrop assessment will receive the same share as another user who has held the same number of tokens for the entire month. This could cause undesirable liquidity squeezes on HBT, which would detract from its main function as a currency. Under the balance-block model, the second user would receive a much greater share of the airdrop relative to the first. This additional share is proportional to the duration and magnitude of their holdings, thirty times more in this case, as they would have held 100 HBT for thirty days compared to the 100 HBT held for one day by the first user.

Airdropped NII will be distributed to users in accordance with their balance-block holdings over the qualifying period. While this method of distribution will serve to minimise monthly liquidity squeezes on HBT trading due to the periodic nature of the airdrop, we have also chosen to utilise balance-blocks as we strongly believe that this form of distribution is the fairest and safest possible way to organise an airdrop. The Nahmii airdrop uses the Nahmii protocol to deliver tokens directly to a user's off-chain wallet. Unlike with most airdrops, which must deliver a minimum value of tokens in order to be viable, Nahmii's flexible fee structure means that no minimum holding is required to participate in the airdrop. This feature highlights a key benefit of using Nahmii: micropayments are now viable, as the fee for sending these payments is commensurate with their value.

Note that the distribution of tokens during the Nahmii airdrop has not been a trustless process so far due to the limitations of the Ethereum network. While the airdrop is currently processed by Nahmii AS directly, this task may be handled by the Nahmii Foundation in the future. There is also a possibility of making this process more trustless in Nahmii with generalised smart contracts support.



Nahmii