



Palliora

A Decentralized Platform For Sharing Intelligent Thoughts

Version 1.1

Carlo Brunetta* Amit Chaudhary[†] Stefano Galatolo[‡] Massimiliano Sala[§]

March 2, 2026

Abstract

Human society is transitioning to a knowledge economy. The rise of Artificial Intelligence (AI) has once more shown that the value of data grows dramatically as they are shared and aggregated. However, from established applications like Finance, which require selective handling of confidential information, to the AI revolution - the same issue persists: we share data with centralized entities with no guarantees on the exact handling of our data. We propose a system (Palliora) based on a public blockchain in which everyone can selectively share their data for trusted, confidential and verifiable computation, knowing that only the intended recipients will have access. Moreover, such a system guarantees fairness to all participants in an AI ecosystem: data providers, model providers, training providers, model users, which is essential for an unplugged growth of AI technology. We achieve our goals by introducing in Palliora both several independent actors, that populate overlapping networks, and a peculiar contract standardization. Among our actors we have DA nodes, which take care of Data Availability, Guardians, which take care of a decentralized access-control system, Calculators, which perform outsourced computations (with varying confidentiality levels), and Verifiers, which check the quality and consistency of input and output data. Palliora's contracts permit free negotiations among all users and all actors, while enforcing correct execution and fair distribution of rewards.

Acknowledgments

This whitepaper has been reviewed by Professor Nigel Smart (wikipedia page and web page) and Professor Massimo Morini (Google Scholar and LinkedIn).

*ShunyaXifra, cbrunetta@shunyx.net, [orcidID](#)

[†]University of Warwick, UK, amit.chaudhary.3@warwick.ac.uk, Palliora, amit@palliora.org, [orcidID](#)

[‡]Università di Pisa, Italy, stefano.galatolo@unipi.it, [orcidID](#)

[§]University of Trento, Italy, massimiliano.sala@unitn.it, ShunyaXifra, msala@shunyx.net, [orcidID](#)

Contents

1	Introduction	3
1.1	History and Defining the Problem	3
1.2	Addressing the Problem	5
1.3	Our Contributions	5
2	Palliora’s Design	7
2.1	Palliora Overview	7
2.2	Data Availability Layer	9
2.3	Computable Contracts	12
2.4	Guardians Network	13
2.5	DAA and DAC Logic	19
2.6	Tokenomics and Rewarding Mechanisms	21
2.7	Interaction with the Outside: Bridges/Oracles/dApps	26
2.8	Palliora Example Functionalities Instantiation	27
3	Palliora’s Applications	30
3.1	Finance	30
3.2	Privacy in blockchain execution	30
3.3	AI Application	31
3.4	Other Applications	31
4	Details of AI Applications	32
4.1	Open Source AI	32
4.2	Encrypted Models for AI	33
4.3	AI Marketplace	34
4.4	Verifiable and Decentralized AI Agents	35
4.5	Related Works	36
5	Remarks and Comparisons	37
5.1	Remarks	37
5.2	Available Implementations	40
5.3	Conclusions	44
A	Stabilization Mechanism for the Staking Rate	50
A.1	Simulation Source Code	53

1 Introduction

Human society is transitioning to a knowledge economy. We mostly rely on specific institutions to safeguard our data and perform meaningful computations on our valuable information. The current system needs intermediaries that must be trusted, with limited possibilities of external independent verification. From established applications like Finance, which require confidential information to be stored, shared and processed, to the AI revolution - the same issue persists. We share data with centralized entities, but we cannot be sure of the exact handling of our data. As a consequence, in the current system we may fear a loss of confidentiality and privacy. This is unfortunate, because reluctant data owners cannot benefit from the increased value of their data through computations that would result from their sharing. In our opinion, this large-scale trust-based intermediation of data and computation introduces *significant* inefficiencies in the knowledge economy.

Many digital platforms offer services that require users to share data in order to perform cloud computations, without meaningful user-side verifiability. The possibility of information leakages and loss of privacy not only discourages people from using the services, but it often leads them to use isolated environments that are costly to maintain, very inefficient and prone to security incidents.

This problem is worsened in AI scenarios. On one hand, AI capabilities have revolutionised human-machine interaction by accelerating the automation of tasks (driverless cars, games - chess or Go), medical discoveries (protein folding, gene markings), and human-like assistance (chatGPT, and other models). On the other hand, the marginal value of data increases with aggregation, because only when substantial computation is carried out can interesting models be discovered. Therefore, main AI actors such as data providers, model providers, and computation providers lack a coordination mechanism that is verifiable. Moreover, this mechanism should be suitable for monetisation, that is, all actors get fair rewards for their contribution, otherwise the coordination effort will fail.

Data providers are concerned with the integrity and authentication of their data, the usage of their data (data providers may have ethical, communal, or other restraints), and their confidentiality. Similarly, model providers are concerned with the integrity and authentication of their models, as well as their usage. While most models are public, some providers may want to keep their models confidential. Finally, computation/training providers require integrity, authentication (and some confidentiality) of the input and output of their computation/training. Besides, all actors may wish to delegate the heaviest computations to third parties, still retaining all previous security requirements.

Extending the fundamental AI use case to an agentic economy will exacerbate the problem, as AI agents lack trustworthiness in managing personal data or preferences for end users, while the end users themselves want AI agents to be accountable for their decisions.

We can thus summarise the main problem we aim to address with Palliora, which we believe must be solved to enable an unimpeded knowledge economy and AI development.

To let everyone selectively share their data for trusted, confidential, and verifiable computation, knowing that only the intended recipients will have access to it.

1.1 History and Defining the Problem

To understand how we plan to address our problem, we describe some other problems and see how their solutions actually bring us, step by step, closer to our goal.

First, let us suppose that a group of people wants to make their own digital currency, but in such a way that they do not need to trust anyone to keep their balances (or, equivalently, their transaction history). Therefore, they need a (digital) location where balances are kept *immutable* (to tampering) and where the currency's users can check them freely. Certainly, they need Internet and they need cryptography, but how to put all pieces together is the real question. After years of research, this problem was finally solved by blockchain technology with the introduction of "public ledgers" [14, 63].

The main point in public ledgers that is relevant for us (and that makes people trust a public ledger) is the so-called *economic security* [22, 30, 54], as follows:

for the rationally-behaving system providers (miners and such), it is much more convenient (economically speaking) to follow the protocol rather than to cheat.

Second, the same group of people with a public ledger want to use some data on their ledger as input to perform computations, but in a way that the sequence of operations is executed exactly as planned. Indeed, they may upload a precise description of their program (*smart contract*) on the (immutable) ledger to avoid ambiguity. Still, *system providers must be driven via economic incentives to execute it correctly*. This problem was partially solved in Ethereum [36], even if with a substantial computational cost that led quickly to saturation and scalability problems. A more satisfying solution was found with “rollups” e.g. Arbitrum [52], that is, with actors collecting requests sent to the (smart contract) ledger and afterwards sending a merged computation output to the ledger (possibly accompanied by proofs of its correctness). Again, *these actors follow the prescribed protocol or incur economic penalties*. Yet, different rollups [39, 47, 85, 87] (zk rollups, optimistic and others) provide different levels of assurance, with higher levels being more expensive and less efficient.

Finally, people realised that any kind of data, in principle, could be stored in a way immutable to tampering, not only inputs and outputs of programs. However, such a system must let people retrieve their data at will, as well as provide proofs of availability. Once more, solutions have been found (Lazy Ledger [10], Storj [91], FileCoin [74]) under the generic name of *data availability* (DA), where the providers of DA platforms [8, 24, 35] *either follow the prescribed protocol or lose their economic reward*. Alternative solutions also try to include delegated computation, such as [31].

Having quickly reviewed past solutions, it is clear that any successful solution to our problem (which also includes data privacy) must provide a level of trust to the data owners similar to the level of trust assigned to a public blockchain, a smart-contract platform or a DA system. In other words, the data owners must feel that their data will not be disclosed outside the scope for which they are sharing it. This level of trust is not enough, high as it is, for the general computation scenario, where the so-called confidential computation¹ must also be guaranteed, because several of its participants cannot afford any data leakage.

As a consequence, our solution must be designed so that *if its system providers (who handle the data shared by their owners) disclose and compute data in an agreed-upon way (including confidential computation, when requested), then they are rewarded; otherwise, they are punished (in economic terms)*. This provides the level of *technological* trust appropriate to both the general computation and AI contexts.

Yet, for an AI ecosystem to thrive, another essential feature is missing. With this, we mean that the money flowing into the ecosystem (which may ultimately come from the end users of a trained model) must be distributed following a *fair value attribution* (which we might see as a form of “economic trust”). For example, data providers will need to be rewarded for their contribution to the model training in a way which has to be simultaneously: proportionate to the success of the model and (felt) fair for the data providers. As another example, model providers are actually of at least three distinct categories: those researching foundational and open models, those developing new models starting from the others, those evaluating and testing known models. All of them play a role in obtaining an efficient and valuable final model, so each must have somehow a share of the resulting value.

While we design Palliora with technological trust as the main system feature, we do not include fair value attribution as a system feature in Palliora, for two reasons. The first is that obtaining an objective formula for fair value attribution is very difficult, with too many factors playing and so many possible situations that can occur unexpectedly. The second is that we prefer all Palliora users to interact financially with each other freely, so that natural economic competition will push towards a

¹Despite the terminology is used mainly in the trusted execution environment scenario, we use it as a definition for any computation requiring some level of confidentiality guarantee defined by the context.

(hopefully) reasonably fair value attribution. However, the level of technological trust that we reach with Palliora is itself a guarantee that economic fairness can be reached (e.g., intellectual property cannot be stolen, data cannot be tampered with, confidential computation protects from leakage). It will be the duty of Palliora’s decentralized applications (dApps) to interact with users with acceptable and sustainable economic conditions (e.g., a dApp might collect data from some users and reward them with a one-shot payment, while another will reward them with a recurrent payment proportionate to the model’s usage).

1.2 Addressing the Problem

We address the problem of sharing data (possibly sensitive) in a secure and reliable way so that they can be computed trustworthily, with both general purpose computations and AI applications in mind. At the same time, the computation outputs (and/or the original data themselves) are selectively disclosed only to the correctly designed entities. Therefore, we design and propose a system called Palliora based on an *ad hoc* public ledger that allows the following functionalities.

1. Any user can submit any type of data, along with detailed instructions dictating who can access these data and how; conversely, any user can request any type of data.
2. Any data submission can be accompanied by a detailed description of any computation to be performed (including precise access-control rules governing input and output); conversely, any request of input data can be accompanied by a detailed description of any computation to be performed (access-control included).
3. Any computation can also be accompanied by input verification and/or output verification, according to tests and verifiers agreed upon by users.
4. Any computation can also be accompanied by a formally-defined required level of confidentiality, such as homomorphic encryption or trusted execution environment (TEE) computations.
5. Each actor will receive a fair reward for its computational work and its active participation, whose value is determined partially by public algorithms and partially by direct negotiations (depending on the considered scenarios).

Palliora’s architecture is flexible and general, thus our terminology should be understood in a broad sense. The abstraction of the terms “user” or “actors” allows dApps to utilise Palliora as “users/actors”, while opening the door to numerous applications.

1.3 Our Contributions

In this white paper we propose Palliora, a system that enables all functionalities as described in Section 1.2 within a decentralized framework, based on a public blockchain, providing general-purpose computations specialized for AI applications.

The technical core of this paper is Section 2. After offering a high-level overview of Palliora, we describe Palliora’s own Data Availability layer and its properties. Then, we introduce the “guardians network”, which provides the confidentiality and privacy guarantees to Palliora. Guardians receive the (encrypted) user data and process it, deciding *when* and *to whom* to disclose the output of the corresponding computation. Guardians are rewarded or penalised according to how closely they follow the instructions defined by the users. Guardians include specialized servers for computation execution, computation verification (input and output verification), with varying degree of computation confidentiality. Once the main actors have been presented, we formalize the Palliora transaction format, which specify the roles of parties in a protocol execution. Transactions are inherently linked to “Palliora contracts”, which are presented within our tokenomics model. Palliora contracts are a kind of computable contracts that coordinate the outsourcing of computation and verification, while specifying economical rewards for all participants in the contract execution, not only calculators and verifiers, but also, for

instance, input-data providers. We highlight the link between Palliora’s contracts (negotiated among peers) and macroeconomics. Section 2 concludes with Palliora’s inside/outside relation.

We describe two application fields for Palliora in Section 3, that is, Finance (compliant and private transactions, credit assessment and market design while preserving privacy) and privacy in blockchain execution. These are followed by several AI applications, including *Open Source AI*, *Encrypted Models for AI*, *AI Marketplace* and *Verifiable and Decentralized AI Agents*. With *Open Source AI* we denote a research scenario where many independent AI developers improve on each others’ models, until one model reaches a desired quality target. With *Encrypted Models for AI* we denote a business scenario where a company possesses a large amount of data in need of an AI model that best fits its dataset, but model developers are unwilling to share their models. With *AI Marketplace* we mean that a network of Palliora dApps could act as a huge *marketplace* where all AI actors interact and make deals, such as input providers, model providers, model tuners, etc. With *Verifiable and Decentralized AI Agents* we mean a business scenario where agent developers can deploy agents that remain private and verifiable, simultaneously handling users’ personal data according to the users’s predefined logic and ensuring various levels of confidentiality. Section 3 concludes with a hint to some other possible non-AI applications, in order to showcase the generality of Palliora: Digital Sovereignty (for secret sharing of citizens’ data across public environments), Digital Platforms Advertising: (digital advertising with a focus on privacy-safe options), Health Care and Life sciences (Sharing of personal health data, even with wearable devices).

In Section 4 we detail the most interesting AI applications for Palliora, starting from the problem statement and arriving at the description of the problem solution leveraging on Palliora capabilities. To put this into the right context, we briefly compares Palliora with related works.

In Section 5 we provide some remarks on Palliora’s design, on its security and on some research paths that could be followed to improve Palliora. Before drawing our conclusions, this section describes Manas, i.e. our available implementation of Palliora, including computational efficiency and infrastructure requirements.

Our white paper ends with an appendix, Section A, where we have decided to focus on our design choices related to Palliora’s staking, because this “economic security” part is fundamental for Palliora’s bootstrap and acceptance.

2 Palliora's Design

The following section details a technical description of Palliora's protocol, on which the entire Palliora framework is built.

In Subsection 2.1 we provide a high-level overview of the framework, clarifying the role of the actors, their interaction and the protocol functionalities. In Subsection 2.2 we focus on a fundamental component of Palliora: its own Data Availability layer and its properties. In Subsection 2.3 we focus on an innovative component of Palliora: its computable contracts, their negotiation and their enforcement. In Subsection 2.4 we introduce some key actors of Palliora, called the "guardians". They comprise specialized servers for computation execution and computation verification (input and output verification), with varying degrees of computation confidentiality. In Subsection 2.5 we describe the formalization of the exact roles of parties in a protocol execution, as written down in the Palliora transaction format. In Subsection 2.6 we detail the economic model within Palliora, presented as a tokenomics scenario. Notably, here we explain the link between Palliora's contracts (whose conditions are privately negotiated among peers) and macroeconomics (staking and token generation). In Subsection 2.7 we delve into the interaction between the Palliora framework and the outside. Finally, in Subsection 2.8 we provide an example of how Palliora could be used for real-life applications.

2.1 Palliora Overview

The Palliora protocol is at the heart of a general framework to coordinate the storage, retrieval and computation on sensitive data via a token-oriented reward mechanism that promotes the honest behaviour of all protocol's participants. Palliora's token is called PALI.

The Palliora protocol includes several actor sets:

- ▷ **publisher**, denoted with $\mathcal{P}$, is the actor that provides the initial data/function in the protocol, e.g. a data owner (willing to upload their data for others or willing to process its data);
- ▷ **guardians network** is a network of actors, called guardians. Collectively, they coordinate the Palliora protocol's execution and verify its correctness. Individually, each can play a specific role in computational support. More specifically, there are three possible types of guardians:
 - **a standard guardian**, denoted as $\mathcal{G}$, acts jointly with the other standard guardians to ensure that data are stored/retrieved/computed according to the transaction specification, e.g. these guardians interact to setup the framework to allow the train a model on the publishers' data;
 - **an I/O verifier**, denoted as $\mathcal{V}$, is a specialized guardian with the role of validating the correctness of data in input/output (to the computation required in the transaction) according to some provided tests;
 - **a calculator**, denoted as $\mathcal{C}$, is a guardian that provides computational service and allows the execution of calculations on the publisher's data.
- ▷ **data availability (DA) network**, where each node ("DA node") is denoted as $\mathcal{D}_i$, handles all data management and storing while providing verifiable availability guarantees. The DA layer is composed of DA miners² that manage the DA's blockchain. The DA network might also contain storage nodes specialized in effectively storing huge amount of data, or even other types of DA nodes, but of essence for Palliora is that the interactions between the guardians and the DA network are performed through the DA miners.
- ▷ **retriever**, denoted as $\mathcal{R}$, is the actor requesting access to the guardian network for the original publisher's data, or a derived computation on it, e.g. an aggregated statistical evaluation.

²Notably, Palliora's DA miners are effectively proof-of-stake block validators but, to avoid confusions with the role of the previously described verifiers when explaining the Palliora protocol, we use the "miner"'s nomenclature.

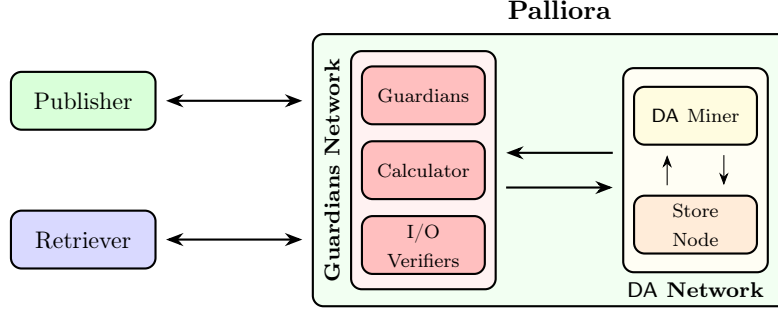


Figure 1: High-level overview of the Palliora entities and their interaction.

Despite our actors' theoretical separation based on the considered Palliora role, a guardian $\mathcal{G}_i$ might cover multiple roles, e.g. $\mathcal{G}_i$ can provide services as both a calculator $\mathcal{C}_i$ and a verifier $\mathcal{V}_i$. Clearly, these multiple roles must be taken in account when considering data security.

The *Palliora contracts* are the core of the Palliora protocol. Given a specific instance of protocol execution, its contract describes:

- ▷ which actors are involved,
- ▷ the role of each actor in detail,
- ▷ which data is provided as input, as well as the exact means by which its correctness is verified,
- ▷ which execution is expected on the input, including its confidentiality level,
- ▷ how the computation is verified,
- ▷ rewards for all involved actors.

All actors agree on an initial Palliora contract fully describing the Palliora application to execute. Further details on the contract's fields are provided in Section 2.5. Once agreed, the contract is fixed (i.e., it is written *immutable*) in the DA layer's blockchain. The contract is later executed by the actors following the contained instructions. By providing correct signatures, positive input/output validations and other details (as per the contract itself), the rewards promised in the contract are effectively released, so that all actors are paid for their efforts.

Palliora's contract are an ad-hoc instantiation of *computable contracts*, i.e. a contractual agreement stored in the blockchain which facilitates the outsourcing of verifiable computation between untrusted parties. Seminal ideas have appeared in [26] while additional information on computable contracts are available in [17, 20].

2.1.1 General Remarks

A first general remark on Palliora is that the symbiosis between the DA layer and the guardians is essential for its working, each playing a complementary role in the protocol. The DA layer provides *data availability* guarantees, i.e. the publisher's stored data can be publicly proved available by the DA layer to any entity requesting availability guarantees. The network of guardians acts as a gatekeeper that allows (or not) access for retrieving or computing on the available data. Additionally, a subset of trusted guardians plays the essential role of verifying that each step is correctly executed, e.g. the data is well-formatted or the computations are correctly executed. Collectively, all guardians (each in its role) support an infrastructure handling publisher's data (which are at the core of any data-oriented application in AI), by provably storing them, limiting access to them, computing them and releasing the related output. In general, the Palliora actors interact while requiring specific security guarantees:

- ▷ publishers require privacy and confidentiality on their data and only allowed actor can access it or some derivative computation;

- ▷ all parties desire an economical reward for their participation according to their role, e.g. a provider buying and reselling agglomerated data from different publishers should pay for the publishers data and the Palliora network fees;
- ▷ data access shall be restricted to only the authorized parties, decided by the publishers;
- ▷ publishers and retrievers require confidentiality, computational correctness and data integrity guarantees during the data handling by the guardians, calculators and DA nodes;
- ▷ both the guardians and DA layer require their computational/responsibility efforts to be fairly (economically) awarded.

A second general remark regards the management of the Palliora project itself. We envision Palliora's technical implementation (e.g. the guardian's source code) and general tokenomics parameters (fees, inflation, similar) to be coordinated via a decentralized governance which manages the requests for improvements from the stakeholders/community and, after either a decentralized voting or other similar consensus phase, the foundation finalizes the request by letting the developer the introduction of the improvements in the Palliora code. Further details on how such type of governance should handle the tokenomics are provided in Section 2.6.

2.2 Data Availability Layer

A fundamental framework underlying the Palliora protocol is the *data availability layer* (DA) that plays the role of a file system by providing storing place for the data/models on a fully decentralized, resilient to network-nodes failures network [8, 10, 24, 35, 61, 74, 83]. Palliora is designed to be instantiated on an ad-hoc DA layer that provides DA guarantees, i.e. a provably secure methodology to publicly verify the availability of the stored data. The DA layer is an internal part of Palliora and not an external service. While from a pure functional point of view Palliora might be thought as a Layer-2 on any DA blockchain, we have decided not to follow this road because of our security models:

- ▶ our security comes from a careful combination of economic security and cryptographic security, therefore our tokenomics requires a close link between rewards at the DA level and at the Palliora contract level, in particular both DA miners and Palliora's guardians will stake the same token,
- ▶ we need our more specialized nodes, notably the so-called "guardians", to interact natively with the DA miners, to minimize possible security threats (mainly, programming flaws) arising from the interaction with a network that we do not control and whose design can change rapidly (also, we avoid the overhead in the communication and monitoring protocol with said network).

In this section, we provide an overview on DA layers, then we provide an explanation on the layer's functionalities and how these relate to the Palliora's DA requirements. Finally, we show a general overview of the tokenomics³ considered in the DA layer. We based our explanation on a survey on storage systems and data availability [21] which can be consulted for more information on the topic.

The DA layer is generally defined by a network of nodes with the goal of providing a distributed storage service that is resilient both against the passage of time and against malicious corruption (or dropping off) of the nodes in the network [1, 8, 10, 24, 55, 74, 83, 90, 91, 92, 97]. Such functionality can be easily achieved by a centralized solution where a set of parties can fully coordinate where data is stored and how it is retrieved. However, a decentralized solution requires additional procedures to handle the lack of trust between the nodes. This can be achieved by considering a blockchain for reaching consensus and maintaining a shared view on the current system status together with a *proof-of-storage* (PoS) [7, 9, 94, 95] or *proof-of-retrievability* (PoR) [79] primitive.

A PoS/PoR is a primitive that allows a storing node to prove that some data m is currently being stored, by providing some form of formal proof π , which can be verified (either publicly or privately)

³Further details on the tokenomics can be found in Section 2.6.

by a querying party. More precisely, if the proof is valid, m is stored, it has not been tampered with and it's retrievable.

Data availability provided by PoS/PoR might achieve flexible properties, such as allowing the proof verification to a selected set of parties or allowing partial verification of stored data [5, 6, 11, 12, 48, 60, 98]. We are interested in solutions that also achieve *data immutability*, i.e. solutions that guarantee that the stored and provably-available data was not modified at any time.

Different PoS/PoR instantiations might be implemented into Palliora, for example a commonly used technique to construct a PoS/PoR is to use a polynomial commitment scheme, such as Kate et al. [53]. Conceptually, data is encoded into a polynomial (either as representation or evaluation) of which a commitment is publicly stored on a blockchain, for immutability. The availability proof consists in providing an evaluation of the committed polynomial on an unpredictable challenge-point. If the evaluation is correct, it implies that the polynomial is stored from which the data can be decoded and retrieved.

Let us consider an abstract view on the process of storing data in the DA layer and consider a simplified tokenomics description. We defer to Section 2.6 for the precise tokenomics description. We use a generic term “user” indicating any actor, program or computer that wants to use the DA layer. The user is interested in one specific data chunk that we refer as “its data”.

As depicted in Figure 2, whenever a user wants to store its data m , the user computes the storing fees via a *fee estimation tool*, provided and managed by the DA layer. After evaluating the fees and possibly adding tips, the user can provide its data to the DA layer where a storage node will save m and the DA miners will record the transaction into a new block. We have thus obtained data traceability. The DA miners and possibly the storing nodes can collect some of the fees for their work.

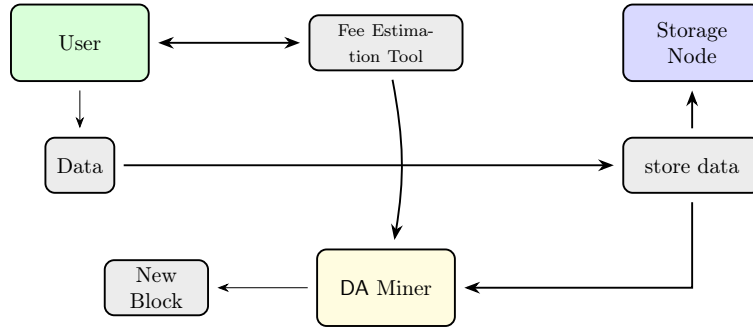


Figure 2: High-level overview of the flow in the DA layer for storing data into a storage node.

At any point in time, the user might request to verify that its data is currently being stored. This can be done by requesting a PoS proof as a challenge for the storing node, see Figure 3. To promote honest behaviour and active participation, the inability to answer lets the storing node incur in a financial penalty (such as the loss of the agreed reward or of a deposit). These mechanisms might be automatized via smart-contracts on the DA's blockchain or other network's procedures.

Whenever the user wants to retrieve m , it interacts with the storing node to retrieve its data. The DA miner logs the exchange on the DA's blockchain to allow the correct distribution of the rewards, see Figure 4. In summary, all actors in the network are paid for their efforts by the user: for the storage, the DA guarantees and the retrievability.

Returning to Palliora, the Palliora DA blockchain provides a token used for rewarding all Palliora actors, including the DA nodes. These tokens are designed as exchanged currency between Palliora actors: publishers, calculators, guardians, I/O verifiers, retrievers and DA nodes.

As common in DA blockchain, Palliora adopts a stake-based consensus. Therefore, nodes must stake an (algorithmically decided) amount of tokens to participate in proposing the next block. DA miners collect the transactions' fees, as well as further rewards/token for their service, and, in case of malicious

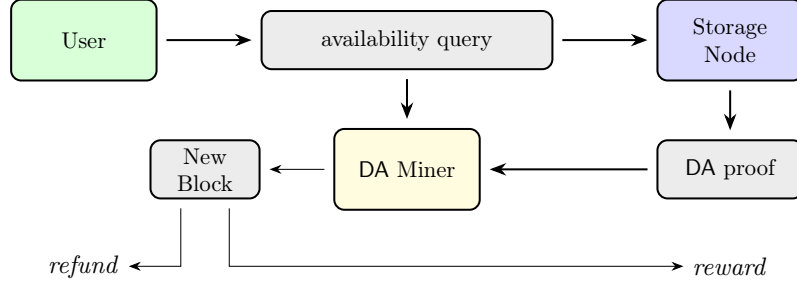


Figure 3: High-level overview in a DA layer for the query of data availability.

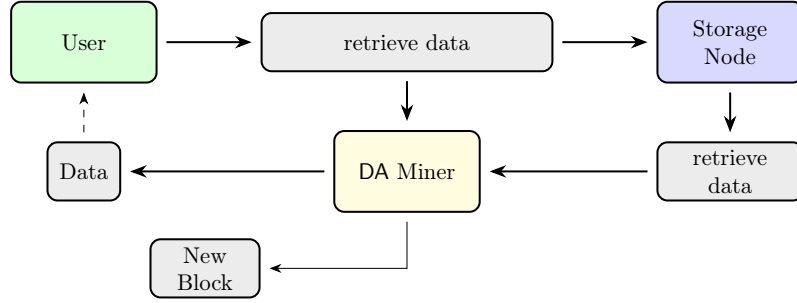


Figure 4: High-level overview of the data flow to retrieve the data from the DA layer. The miner records the request and the correct retrieval of data, allowing the agreed rewarding of the storage node.

behaviour, their stake is slashed (thus economically forcing a DA miner to behave honestly). Palliora consensus protocol is designed to prevent centralization of power in few DA miners.

We remark that each DA solution might differ from this simplified explanation to allow for additional protocol's properties and features that are designed for specific applications. When confidentiality/privacy on data is required, most DA layers delegate these requirements to other actors, which perform preparatory steps (before the upload) in which where data is encrypted using suitable encryption scheme. However, Palliora DA has been designed to satisfy multiple requirements specific for Palliora, therefore it cannot be replaced by another existing DA layer.

Palliora's Implementation. Palliora's implementation Manas⁴ is based on a modified version of Substrate [86], from which Manas inherits the consensus protocol while introducing features to properly handle the guardians' transactions (as described in Section 2.4). To be more precise, data to be stored on Manas DA layer is encrypted and then encoded (using an erasure code), to provide both confidentiality and resilience/integrity. The encrypted/encoded data is represented in a matrix format for which the entries of each row or column are the coefficients of a polynomial used in the Kate et al. [53] polynomial commitment scheme. The commitments of these polynomials are stored in the blockchain guaranteeing immutability.

To prove availability, the DA node receives a challenge in the form of a request to evaluate a stored polynomial. The evaluation is performed on a point chosen by the user and the DA node provides a proof of correct computation which anyone can publicly verify.⁵ By (correctly) verifying the evaluation of some polynomials, the user is reasonably sure has guarantees that the data is still available. Additionally, by cleverly designing the matrix representation of the data, the protocol allows the verification of partial

⁴Manas is described in Section 5.2.

⁵If the node can provide valid evaluation and proof, then the node must possess the correct polynomial representing a specific piece of the encrypted data.

amount of data by requiring the verification of only a (specific) subset of the committed polynomials. However, Manas DA implementation is currently diverging more and more from Substrate, following active research to specialise the DA layer to Palliora's goals.

2.3 Computable Contracts

A key concept in Palliora is *computable contract*, contract from now on. Palliora's contracts are a concrete instantiation of the approach explained in the papers [17, 20] where more research-oriented details are provided. We report and adapt to our needs the approach to allow full understanding on how these contracts are considered in Palliora. Further remarks on Palliora's contract that requires full understanding of the whole framework are provided in Section 5.1.1.

Computable contracts provide a data-structure where actors can agree on the outsourced computations to be made by defining all actors' roles, from the servers performing computations, to those providing input, to those performing verifications (on input and on output), including all related communications. The contracts fully describe the (verifiable) computational interaction, rather than focussing on the computation itself. This is done by coordinating all actors, that do not trust (fully) each other, on a deterministic data-handling procedure which includes fair rewarding for their data/services. The contracts are funded according to the use-case (e.g. retrievers pay if they want to obtain data or a service) where the party funding the contract decides the verifiers, and the verdict of the verifiers is final, but two aspects need to put this draconian description into the right perspective. First, no calculator is forced to accept the user's contract (which crucially contains also the identity of the verifiers), so the founder's apparently-absolute freedom must match the equally-absolute freedom of the calculator. They will find a middle point in their own off-chain discussion, or the contract will not be finalized. Second, the contract specifies also the method used by the verifiers to reach their verdict. It could be anything: one sole verifier trusted by all actors thanks to its reputation, a set of commonly-trusted verifiers taking a decision with a quorum system, an external protocol based on challenge-response, etc. Again, whatever the method, it is only *proposed* by the user, since it must be accepted by all actors or the contract will not execute.

When we say that a party *funds* the contract, we mean that the party commits a quantity of the native cryptocurrency used by the underlying blockchain (i.e. PALI). There is no automatic determination of the reward of the contract's actors since this must be done by reaching an agreement off-chain before the contract is created and provided to the network.

When we write about "actors/users", note that we do not mean solely humans, but it could be any software that follows the protocol, including AI agents.

Our solution is built over a blockchain to inherit the underlying payment method and an immutable public ledger, simultaneously facilitating the negotiation phase. Using cryptographic and economic security models, our approach balances robustness and practicality since cryptographic techniques are used to protect data confidentiality and privacy, while economic incentives ensure protocol compliance and discourage misconduct, as malicious actors can be detected and economically penalized. We also consider a *reputation system* (e.g. [46, 49, 75]) in which the trustworthiness of actors can be measured on the basis of their actions. As we later describe in Section 2.4, all Guardians must register themselves into Palliora thus allowing public auditing of their actions, i.e. their participation in the contract executions is publicly recorded on the blockchain, allowing anyone to extract actors' participation in the protocol, historical participation, and reliability.

A reputation system is somehow required to guarantee that Guardians can be trustworthy, especially in our scenario where trust is not easily given. For example, if a malicious verifier $\mathcal{V}$ falsely claims some verification to be negative, then the relevant parties will be not rewarded. To prevent this, our contract's design may require the data provider to select $\mathcal{V}$ based on the perceived reputation. Moreover, the contract may require a formal proof for the verifiers' honest behaviour, e.g. by forcing the usage of secure cryptographic primitives, or the proof might be constructed by reaching a verifiers' majority (or even more complex quorum situations) where many verifiers are required. In any case, a correctly instantiated contract would provide a mechanism to identify such misbehaviour, thus forcing an economic penalty to the malicious $\mathcal{V}$.

The largest part of the computational and communication effort is performed off-chain, i.e. agreeing on the contract's content, better explained in Section 2.5. Afterwards, the contract is effectively encoded as a transaction which must be natively handled by the Palliora underlying blockchain, i.e. contracts are first-class citizen on the blockchain and are effectively verified by the network. This allows us to achieve contracts' decentralized handling and enforce their correct execution via a combination of cryptographic techniques and economic security. More precisely, the contract is encoded as a series of on-chain transactions, allowing the underlying blockchain not only to store the agreed final version of the contract, but also to track all contract's execution.

Contracts define the functions to be evaluated by considering "data/function location" which allow the retrieval of the data/function. Such location can be implemented with a simple web-link or an appropriate oracle call, clearly in relation to the contract's security requirements. Palliora allows both leaving the choice to the contract's requirement, i.e. an oracle call is more complex but provides a higher reliability than a web-link.

The considered functions/algorithms might be deterministic or probabilistic in nature, since such a choice does not modify our protocol's execution in any way. We only require computational termination, which can be obtained with a bounded evaluation time. This highlights the importance of the contract's setup phase, where actors must decide 'what to compute' and highlights 'where' algorithms are effectively stored thus requiring additional attention on the functions agreed upon. Depending on the application, functions might be publicly version-traced or limited to a certified set, e.g. computable functions might be selected as training or evaluate inference from a pre-defined list of AI model.

Another key point of our implementation is that confidentiality guarantees can be tuned according to the contract's requirements, both on data and calculations. In other words, Palliora's contracts provides an adaptable framework for verifiable outsourced computation, with different security requirements, from strict usage of fully homomorphic encryption (FHE), functional encryption (FE) schemes, secure multi-party computation (SMPC) and SNARK/STARK, to techniques that leverage the protocol structure and the penalties for misbehaving. Data security is obtained via surgical usage of appropriate cryptographic tools.

2.4 Guardians Network

The core network defining Palliora is the guardians network that coordinates the parties using the Palliora protocol and acts an intermediary entity when uploading/retrieving data from Palliora's DA layer. This section defines the roles of the guardians in Palliora and their protocol. Additionally, we provide some details on:

- ▷ security requirements related to all confidential-computation scenarios;
- ▷ advanced cryptographic primitives suitable to achieve the corresponding security levels,
- ▷ assumptions for the protocol working; and
- ▷ how the network can be effectively instantiated.

Palliora is bootstrapped by initializing the underlying DA blockchain. For example, this procedure comprehends that the DA miners get involved in the framework by staking their tokens. Once the DA blockchain is fully operative, the guardians can register themselves into Palliora (by creating a traceable data structure which is stored in the chain) and stake an amount of tokens. The guardians' data structure contains all their information required by the Palliora protocol, for example:

- *public key(s)* to correctly authenticate $\mathcal{G}$;
- *staked balance* to denote the stacked amount used as collateral (and economically force the honest protocol's behaviour);
- *total balance* to indicate the amount of rewards and fees collected so far (by providing their Palliora services);

- *guardian's type* to indicate which services are offered by $\mathcal{G}$, i.e. standard guardian, calculator, input verifier or output verifier;
- *active status* to signal if $\mathcal{G}$ is currently active and participating in Palliora.

After providing this (and more if required) information, the DA miners store the guardians' data structures and allow data updates via appropriate authenticated requests, e.g. $\mathcal{G}$ can request to deactivate its status when finishing its tasks by sending an authenticated message to the DA miners (which verify the request and record it in the DA blockchain). In practice, the registration can be executed via an activation queue where guardians join the queue waiting to be registered in Palliora. Clearly, the DA miners finalize the registration only when they can attest that the request satisfies all prerequisites, such as staked balance or waiting time.

As previously introduced, at any point in time, Palliora's network is split between guardians, calculators, input verifiers and output verifiers. Calculators offer their computational services while verifiers take the important role of checking that the protocol is correctly executed: the input verifier checks the publisher's input, while output verifiers control the computations executed by the calculator. These verifications are done by running pre-decided tests. The output of these tests is not necessarily confined to a binary "yes/no", rather the output can be used to calculate some parts of the reward, e.g. when a publisher is providing valuable input to a computation, an input verifier measures the quality of the publisher's data (with a public test) and its output is used as weight for the publisher's reward. In the presence of an inadequate (malicious, inefficient, etc.) verifier, Palliora will economically penalize $\mathcal{V}$ (see Section 5.1). Similar penalties are implemented for any malicious guardian or calculator.

The guardians' protocol main objective is to build an initial transaction involving all relevant entities that fully describes a Palliora's *contract*. This contract describes typical information such as: what data is used, where to find it, who might execute the computation, which computations to execute, who can retrieve the data and under which access rules. Plus, the contract must clarify Palliora's specific information:

- which guardians and verifiers are considered in the contract execution;
- the order of following transactions;
- how to split the reward and costs.

The contract is fully encoded into two parts: a *data availability and accountability* logic (DAA), i.e. who and how data/computations can be accessed via the guardians, and the *data availability and computability* logic (DAC), i.e. what is computed and by whom, where input/output are located and how verifiers should verify each step (see Section 2.5).

To account for dynamic presence in the network, the Palliora protocol relays of threshold oriented properties. In particular, a contract might identify a set of n initial guardians/verifiers (that commit their efforts for the contract-handling) of which a quorum t is effectively required in the contract execution. Such a set of guardians can be specified as a list of trusted guardians/verifiers or can be dynamically obtained from the guardians volunteering in the participation of the contract (after evaluating the rewards).

The correct protocol's execution is ensured by *economic security*, i.e. the only way for a guardian to be economically rewarded is to behave honestly and follow strictly the Palliora protocol. Observe that all data and computations are traceable, since they are contained in transactions and effectively publicly exchanged in the Palliora network, where signatures provide guarantee that Palliora's parties agreed on the contract's execution. Additionally, signatures can be easily verified by anyone, in particular the DA miners that are designed to store the contract's execution in their blockchain.

An important concept used in the protocol is nested layers of data encryption and how these are intertwined to provide confidentiality guarantees with respect to different actors in varying moments of the protocol's execution. Notably, $\mathcal{P}$'s data is prepared by $\mathcal{P}$ via an encryption scheme designed to

guarantee confidentiality and allowing only specific receiving parties to decrypt. The prepared data can be shared with the network by applying another encryption scheme/technique, e.g. secret sharing of the prepared data or threshold key-encapsulation. These security techniques are discussed in the next subsections and are encoded in a field of the contract (see Section 2.5).

Notation. We denote with (Enc, Dec) a public-key encryption scheme⁶, $(\text{Share}, \text{Recon})$ a secret sharing scheme [13, 81] where (t, n) are the required t threshold amount out of n parties, $(\text{Sign}, \text{Ver})$ the underlying blockchain's signature scheme algorithms. We define the key-pair for the entity $\mathcal{X}$ as $(\text{sk}_{\mathcal{X}}, \text{pk}_{\mathcal{X}})$. All the cryptographic primitives used in Palliora must be secure according to the applications required specified in the contract agreed upon by all the Palliora's entities.

2.4.1 Confidential Computability

To initialize the Palliora's execution, a publisher must set up an initial transaction denoting the contract requirements in agreement with other parties, e.g. calculators and retrievers. This contract is completely encoded in a Palliora transaction in a (DAA, DAC) logic form of which full description is provided in Section 2.5. Intuitively, these logics fully describe the whole contract's information such as the cryptographic technical aspects (e.g. crypto-suite, security parameters), rewards for the network, data retrieving and calculation rules and indication for verification steps.

The guardian's protocol can be described into phases: an *initial setup* that creates the Palliora contract and identifies all the required guardians participations, a *sending phase* where data is effectively introduced in Palliora and a *retrieving phase* where data is retrieved where economic agreements are fulfilled. These phases define a single transaction and, by combining multiple transactions which are clearly identified in the initial setup, the protocol can handle more complex contract where multiple data exchanges are required. For example, a single transaction defines the direct exchange from a publisher $\mathcal{P}$ to a retriever $\mathcal{R}$ of some data, while out-sourcing computation requires multiple transactions including the provision to the data to a calculator $\mathcal{C}$ and the retrieval of the output.

The initial setup procedure is executed as follows:

1. $\mathcal{P}$ prepares the preliminary transaction tx_0 with all the relevant (DAA, DAC) fields, especially those necessary for the guardians to evaluate the economical reward in providing their service. At this stage, some fields might be initially masked to guarantee confidentiality/privacy and avoid guardians to discriminatory reject offers solely based on, e.g., the retriever's identity;
2. $\mathcal{P}$ shares the transaction tx_0 to the guardians network. A set of guardians $\{\mathcal{G}_i\}_i$ matching the conditions provided in the contract, checks tx_0 , accepts to handle the contract and signal their willingness by individually signing tx_0 , indicatively $\sigma_i \leftarrow \text{Sign}(\text{sk}_{\mathcal{G}_i}, \text{tx}_0)$;
3. $\mathcal{P}$ collects and verifies the signatures $\{\sigma_i\}_i$ and, after obtaining enough participants, moves to the following phase. If not enough guardians are providing their signatures, tx_0 remains idle in the Palliora mempool until its lifetime expires as indicated in the DAA logic.

At the end of the setup procedure, the parties agree on a Palliora contract describing all the protocol's rewards and fees, plus any details concerning the data flow and the guardians required for the correct contract execution.

After finalizing the contract, the protocol executes a defined sequence of transactions that culminate in providing some derived data to the DA layer (that will be finalized in the blockchain by some DA miner $\mathcal{D}$). This phase, denoted as sending phase, is executed as follows:

1. $\mathcal{P}$ prepares the original data $\mathbf{m}$ to be provided in the transaction according to the contract's specification. We denote as *datum* the prepared data $\mathbf{d}$. To maintain confidentiality, the data is effectively encrypted with the appropriate scheme and according to the required application, e.g.

⁶Or a hybrid encryption scheme, via a key/data encapsulation mechanism (KEM/DEM paradigm) [32].

datum can be computed as the encryption $d \leftarrow \text{Enc}(\text{pk}_{\mathcal{R}}, m)$ of a public key asymmetric scheme using the retriever's public key $\text{pk}_{\mathcal{R}}$ to achieve a designated retriever solution.

Observe that the encryption and preparation of datum is executed by $\mathcal{P}$ and not by the Palliora network;

2. if relevant, $\mathcal{P}$ finalizes the contract fields, e.g. reveals masked identities;
3. for each participating guardian $\{\mathcal{G}_i\}_{i=1}^n$, $\mathcal{P}$ computes the secret sharing of the datum d into shares $(d_1, \dots, d_n)$ and each one is encrypted for the specific guardian as $c_i \leftarrow \text{Enc}(\text{pk}_{\mathcal{G}_i}, d_i)$, following the DAA specifications;
4. starting from the transaction tx_0 , $\mathcal{P}$ creates the new transaction tx_1 containing the previous transaction tx_0 with the additional relevant modifications, all the participating guardians signatures $\{\sigma_i\}_i$ and the encrypted shares $\{c_i\}_i$.
5. $\mathcal{P}$ adds the DA fee which is willing to spend at most, finalizing it into a complete transaction tx_2 that is signed and sent to the Palliora mempool;
6. after the appropriate verification of the transaction's validity (together with possible additional verification on the datum specified by the contract), the volunteering guardians sign tx_2 obtaining the final transaction tx_3 which is sent to the DA's mempool for storing on the DA layer. The guardians store their share and relevant transaction information for a consequent retrieval phase;
7. the DA miners receive tx_3 , verify their validity of the guardians' request and (if the DA fees are fair) the whole transaction is stored in the DA layer. Otherwise, tx_3 remains in the mempool until it eventually expires;
8. as soon as the transaction is contained in a block, $\mathcal{P}$ obtains an identifier for the mined transaction that can be provided to the retriever or calculator to follow-up on the next contract's step.

After the data is stored on the DA layer, the contract can follow up by releasing the datum to the appropriate entity, being it a retriever $\mathcal{R}$ or a calculator $\mathcal{C}$. The main goal for the retrieving phase is for $\mathcal{R}$ to request the access to the datum d by paying the relevant guardians responsible for the contract. More formally, the protocol is executed as:

1. $\mathcal{R}$ creates a request transaction rx_0 indicating:
 - $\mathcal{R}$'s identifier;
 - the datum it wants to retrieve, e.g. by indicating the DA identifier of the transaction tx_3 ;
 - additional information as described in the DAA logic contained in the contract;
 - the lifetime of the request.
- $\mathcal{R}$ adds both the guardians fee and the DA fees required for the retrieval operations;
2. $\mathcal{R}$ signs rx_0 and sends it to the mempool;
3. the guardians online check in their internal storage/system if the transaction identifier specifies their participation in the transaction's contract. If so, they verify that all the contract's logics are fulfilled for the request rx_0 ;
4. If the checks are satisfied, the guardians retrieve the encrypted shares $\{c_i\}_i$ and execute the share reconstruction of the decrypted share $d_i = \text{Dec}(\text{sk}_{\mathcal{G}_i}, c_i)$ and gossip the reconstructed datum d ;
5. The datum d is provided in the DA's mempool as a guardian's transaction. The DA miners verify the correctness of d according to the information provided in tx_3 and rx_0 . If everything is verified, the contract's payment are executed and recorded in the DA's blockchain.

The provided protocol allows the exchange of data from one party to another. To achieve computations, the protocol defines multiple transactions each indicating one data exchange and are all coordinated by a contract execution agreed upon in the initialization phase. More concretely, as represented in Figure 5, the execution of an outsourced computation would be described as:

- a. a transaction tx_a where the publisher's data is prepared as datum and provided to a calculator that is the retriever for this transaction. Observe that the effective retriever $\mathcal{R}$ of the computation's output is listed in the DAC logic;
- b. a request transaction rx_a from the calculator used to retrieve the datum d ;
- c. after the calculations are executed, the calculator prepares the output $f(m)$ into the transaction tx_b for the retriever $\mathcal{R}$;
- d. the retriever $\mathcal{R}$ requests the output $f(m)$ via the transaction rx_b .

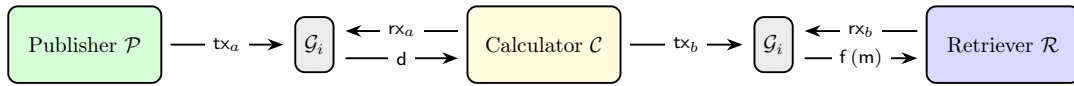


Figure 5: High-level overview of the transaction flow for the execution of a DAC contract where a publisher $\mathcal{P}$ requests a calculator $\mathcal{C}$ to compute a function f and provide the results to a retriever $\mathcal{R}$.

More complicated contracts can be executed on Palliora if they can be described using the (DAA, DAC) logic (Section 2.5). For the remainder of the section, we several different privacy and confidentiality levels (based on the amount of trust provided to the guardians), explain how the verifier is introduced in the guardian's protocol and we detail the crypto-suite requirements.

Trusted Computability. An important aspect to consider in the Palliora's protocol is what we identify in the *trust hierarchy*. For example, the publisher might interact multiple times with an honest guardian $\mathcal{G}$ to the point that it can be considered trustworthy to handle more sensitive data. Providing additional trust into specific actors can simplify the contract's execution and request the usage of simpler cryptographic primitives. A trusted guardian can obtain access to the plaintext data to allow the verification of the data according the contract's indication, before executing the encryption for the designated retriever (without asking an additional verifier and more advanced encryption scheme). Clearly, the efficiency gains must be a weighed against the privacy/security risks introduced according to the application's assumptions.

Whatever the case, the minimal amount of trust considered is that each party acts honestly if economically motivated by the protocol's tokenomics. We later discuss the fully untrusted execution scenario.

An implicit assumption is that the communication between nodes must be confidential, e.g. a TLS connection or similar end-to-end communication protocol, except for public broadcast messages. Regarding the protocol's phases, during the setup phase the protocol requires an encryption scheme to encrypt the data into datum, whose properties are based on the application considered, for example:

- ▷ if $\mathcal{P}$ and $\mathcal{R}$ (or $\mathcal{C}$) have an already established shared secret, then a symmetric encryption scheme can be used to prepare the data into datum;
- ▷ if the contract requires an external verifier $\mathcal{V}$ to verify the content of the datum, then the encryption scheme must guarantee the ability to decrypt the ciphertext to both $\mathcal{R}$ (or $\mathcal{C}$) and $\mathcal{V}$ with separate keys;
- ▷ if $\mathcal{R}$ (or $\mathcal{C}$) is unknown at the moment of data preparation, then the encryption scheme must allow the guardians to execute a proxy re-encryption whenever the designated $\mathcal{R}$ (or $\mathcal{C}$) is indicated by $\mathcal{P}$.

As indicated in the protocol, there are at least two distinct encryption layers used to maintain confidentiality, one towards the networks (guardians and DA layer) and one towards $\mathcal{R}$ or the calculator $\mathcal{C}$. The data preparation guarantees the confidentiality towards any actor except the designated retriever and verifier, if applicable. On the other hand, the second layer provides security guarantees between the networks entities to allow for secure and verifiable execution of the Palliora protocol, e.g. the guardians' shares are re-encrypted to limit the access and postpone the reconstruction only at the protocol's required timing.

The primitives choice must be clearly indicated in the DAA_1 logic in the cipher-suite field to enforce the correct protocol's execution and confidentiality. An appropriate cipher-suite selection permits Palliora to provide fair accountability considering the pseudo-anonymous actors involved in the protocol. However, additional requirements are necessary to guarantee the computational verifiability of the actors' actions while handling data. Confidentiality and verifiability are two distinct trust concepts indicating complementary requirements for the cryptographic primitive to be used, e.g. a guardian might be trusted in handling confidential data, but every computation requires an additional proof of correct computation requiring some sort of verifiable computational scheme or similar.

Untrusted Computability. The Palliora protocol allows the execution of untrusted computability (where no actor is considered secure), thus always requiring data and computation to be confidential and verifiable. These scenarios can be handled by means of FHE [33, 41, 58, 96], FE [2, 15, 59], SMPC [57, 71, 99] protocols or a more technical oriented-solution based on TEE [50, 62, 77]. Each of these solutions can be incorporated in the Palliora contract by filling the appropriate fields and, if additional communication's rounds are required, these will imply additional transaction to encode in the contract. Palliora provides the calculators with the appropriate data and additional parameters required to execute the untrusted computations (which does not affect the guardians' coordination role). On the other hand, any confidential calculator may need to develop its own interface to interact with Palliora, which include the handling of specific values/strings, such as (public, evaluation, wrapped, etc.) keys, shards, parameters for key derivation and so on.

We now provide three examples of untrusted computation.

- the publisher may request the computation of an aggregation function f using FHE, thus requiring all data d_i to be appropriately⁷ encrypted $c_i \leftarrow \text{Enc}(\text{pk}, d_i)$ and, additionally, requiring an evaluation key ek to be provided. The guardians will provide all these inputs to the untrusted calculator that will compute the homomorphic evaluation $c \leftarrow \text{Eval}(\text{ek}, \{c_i\}, f)$. The calculator knows f but it must not know the input/output.
- if the aggregation is executed via an SMPC protocol between multiple calculators $\mathcal{C}_i$, the publisher encrypts each datum $\text{Enc}(\text{pk}_{\mathcal{C}_i}, (d_i, \tau_i))$ together with additional information τ_i required by the SMPC protocol. Each calculator receives from the guardians its own ciphertext which can be decrypted to obtain the datum (d_i, τ_i) together with the required SMPC additional information. The output of the SMPC protocol is provided back to the guardian's network.
- another example would be to use TEE in the following way: the publisher performs the encryption $c_i \leftarrow \text{Enc}(\text{pk}_{\text{TEE}}, d_i)$ where pk_{TEE} is the public key of the TEE. The guardians will pass these inputs c_i to the TEE alongside pk (the public key of the retriever). The TEE will decrypt c_i , compute⁸ f and output the result encrypted with pk .

We could produce much more complicated scenarios (whose description we omit) by adapting the previous previous examples to require that the calculator does not know even f .

As regards computation verifiability, it can be public or private, depending on the usage in Palliora, what matters is that the rewarding cannot be maliciously influenced by $\mathcal{V}$. If it is private, the verifica-

⁷ pk , the publisher's public key, must also be the homomorphic public key corresponding to the FHE computation done by the calculator. However, the calculator does not know the related secret key.

⁸The computation of the TEE calculator will be much faster than its analogue FHE calculator since the computations are done in cleartext.

tion is performed only by the verifier and everybody is supposed to trust its output which (in general) can be very risky. To shun this risk completely, we need a public verification system. This can be implemented cryptographically, according to the verification algorithm to be executed, by e.g. polynomial commitment scheme, zero-knowledge frameworks such as SNARK/STARK [39, 56] or similar verification frameworks.

If public verification is too expensive, another interesting possibility for the verification guarantees is to obtain them by an execution in a (trusted) TEE. Indeed, TEE provides the verifiable execution-correctness of tests, due to its own design.

Economic security can help strengthen the guarantees of correct execution in several ways, for example:

- when the inputs/outputs are private and there is private verification, multiple verifiers can be involved and acceptance/rejection is given by the majority vote; the verifiers which are outnumbered will not get any reward.
- when inputs/output and verification algorithms are public, anyone can check the execution correctness of tests and notify the malicious behaviour; it is up to any publisher to take into account this notification when choosing the guardians.

2.5 DAA and DAC Logic

In this section, we explicitly define the format of the DAA and DAC logics required by Palliora and provide a high-level overview in Figure 6. We list all possible fields of a transaction and provide an explanation on the usage in the Palliora protocol. We split both the logics into two complementary parts, i.e. DAA into (DAA₁, DAA₂) and DAC into (DAC₁, DAC₂), which contains:

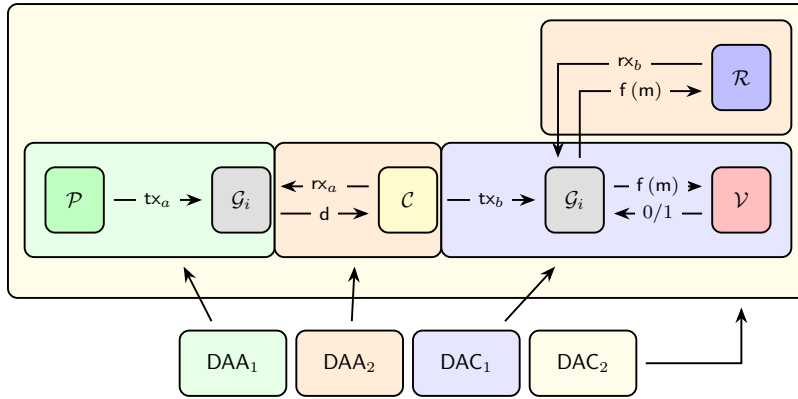


Figure 6: High-level overview of an example of transaction described by (DAA₁, DAA₂, DAC₁, DAC₂).

- ▷ DAA₁ denotes the information required by the standard guardians to understand how to manage the transaction when they first see it, notably including delivering it to the DA layer:
 - *lifetime*, i.e. time-frame for which a transaction is valid;
 - *cipher-suite*, e.g. encryption scheme/protocol the standard guardians must execute;
 - *crypto-parameters*, i.e. parameters of the cipher-suite such as threshold, number of parties, security parameters, etc;
 - *how to select the standard guardians*, e.g. who to trust either as a list to choose from or an empty list (indicating that every standard guardian is trusted)
 - *data digest*, i.e. a commitment/digest of the data for future immutability verification;
 - *standard guardians' fee*, i.e. the amount of rewarding fees to be provided to the standard guardians that correctly executes the contract;

- *additional info on data*, i.e. other information for standard guardians to decide whether to accept the request, e.g. data size.
- ▷ DAA₂ information on the retrieval phase (and unnecessary in the previous phase):
 - *who* can get the data, either as an entity or a threshold. This value might be masked while creating the contract; The retriever identifier must be cryptographically linked to an account;
 - *when* the data can be accessed;
 - *additional conditions* in free format or possibly even oracles. It's possible to consider partial disclosure of the data, for example;
- ▷ DAC₁ describes all the DAA agnostic information related to the computation to be performed, both the inputs/output details, the function to compute and the verification required.
 - *deadline* for the computation execution;
 - *confidentiality level of execution* encoded from a list of possibilities, e.g. '0' for no confidentiality, '1' for TEE (with additional specification required), '2' for FHE (with specifications of algorithms and parameters), '3' for FE (with specifications of algorithms and parameters), etc.
 - *pointer to locations of input*, plus hash of locations/contents. It could be in the current transaction, part of it, or point to something external. Each location could be handled by different entities, what matters is that its content can be decrypted by both the verifier (in principle, one for each location) and the designed calculator (the server performing the computation).
 The contents of the "input locations" includes everything needed to execute the computation, even in the event of more confidential methods, e.g. for FHE, the encrypted input must contain the evaluation keys while for SMPC, the encrypted input contains the individual encryption of each party's input and further computational information;
 - *pointer to location of output* (i.e. where to put it), plus hash of location/contents;
 - *pointer to location of programme/execution* (i.e. what to execute), plus hash of location/-contents;
 - *pointers to location of input verification* (i.e. what to execute to verify the input), plus hash of contents/locations;
 - *pointer to location of output verification* (i.e. what to execute to verify the output), plus hash of contents/locations;
 - *the identity of several actors*, identifier of the actor or could be provided as list. The actors are input provider, input verifier, calculator, output verifier and contract payer;
 - *function (or how to compute) the money/fee for the work executed*. A list of function are provided for the calculation of the fee for the input provider, the input verifier, the calculator and the output verifier. More information in Section 2.6.
- ▷ DAC₂: contains DAA related information which forms the flow for the computational process and can collect all the required signature, similarly to receipt of correct contractual execution.
 - *list of transactions and their role in the contract*. There is a list of transaction denoting how the contract must be executed. These are used to connect the different transaction for the correct rewarding/refunding of the actors involved in the contract;
 - *list of signatures of the entities involved in the contract*. These signatures denote the final agreement of the contract. These signatures are created while generating the transaction to be mined. More specifically, the signatures are from input providers, calculators, verifiers and contract payer.

Depending on the application, several fields in both the logics are optional. The mandatory fields are found in DAA and indicates how data must be stored and how it is retrieved.

2.6 Tokenomics and Rewarding Mechanisms

In this section, we present the internal tokenomics of Palliora, designed to fairly compensate all actors involved in a contract (including the guardian network and work done on the data availability, DA layer) and to actively promote participation and decentralization in the chain.

As regards the *Palliora contracts*, they are the core of the Palliora protocol and are described in Subsection 2.3. In brief, given a specific instance of protocol execution, its contract describes:

- ▷ the involved actors, including their roles,
- ▷ which data is provided as input, as well as its verification,
- ▷ which execution is expected on the input, including its confidentiality level,
- ▷ how the computation is verified and by which actor, including possibly the output and some intermediate internal states,
- ▷ crucially, some **rewards** for all involved actors.

The contract is stored in the DA layer's blockchain and ready to be executed.

As regards Palliora internal tokenomics, Palliora has its native token **PALI**. Tokens will be used to reward the work and the information provided by the various actors in a work/contract, as well as the services of the underlining DA.

We follow the proof-of-stake methodology, but we choose a very specific path to satisfy our complex needs. The system adopts a dynamically programmed inflation rate, designed to adapt in real time to the network's staking conditions. The tokens generated through this inflation mechanism are mainly used to reward participants who stake their tokens and thereby contribute to securing the network⁹, according to a predefined security threshold.

Newly minted tokens are mostly distributed to DA miners and to guardians, incentivizing alignment with the network's target security budget.

A subtle matter is to balance rewards coming from the general security of the system (mainly, from the inflation) with those coming from the handling of contracts. Indeed, given the wide range of agreements and potential collaborations that can take place within the Palliora framework, the rewards for the actors participating in a specific Palliora task¹⁰ is more efficiently determined in a negotiation by the participants (see e.g. Section 4.3) within a contract signed by all participants involved in the task. The tokenomics aspects of these contracts will be discussed in detail in Section 2.6.1.

Another subtle matter is that the above-mentioned computational work need to interact with the DA layer, producing transactions, data storing and retrieving, again with the due privacy constraints. The actors involved in this part of the work will be specifically remunerated similarly to other modern DA services. This is detailed in Section 2.6.2.

As it is usual in PoS systems, Palliora staking yield derives from two main sources: inflationary issuance of new tokens and the redistribution of DA transaction fees. The share of fees contributing to staking yield is modulated based on system conditions and is related to the current staking rate, compensating stakers more when overall participation is low and less when staking is abundant.

In an initial phase, the percentage of newly minted tokens distributed to DA miners is set to $\frac{2}{3}$, while $\frac{1}{3}$ is distributed to active guardians proportionally to their staking¹¹. These percentages are periodically reviewed by the Palliora governance with the aim of maintaining a healthy balance in the composition of the Palliora agents. In particular, Palliora will employ a *Fee Pricing Committee (FPC)* responsible for establishing and revising fees and rewards within the network. The FPC's objectives include: ensuring equitable fees that promote broad participation; maintaining network stability; encouraging experimentation and growth within the ecosystem. The FPC evaluates periodically the economic

⁹This approach encourages decentralized participation and promotes network security, ensured by economic collateral.

¹⁰E.g., computation, verification, executed within Palliora under appropriate privacy constraints.

¹¹To promote their participation to the network

security and network revenue to derive stable fees in real terms. Proposals for fee adjustments are submitted to the Palliora governance, which evaluates them and, upon reaching a majority consensus, implements the changes.

Palliora inflation ranges between 3% and 6%, with maximum inflation reached when the staking rate is between 50% and 60%. This setup ensures that the system dynamically balances token issuance with economic incentives, discouraging under-staking or over-staking. Since Palliora and its token are intended as tools to facilitate the organization and implementation of an information technology service system, the stability and the safety of the system itself are a priority. Our particular implementation of the dynamical inflation and the associated staking reward system aims at ensuring the stability of the staking rate around the optimal one. The way this is implemented is explained in detail in the Section A, while we provide a broader analysis of the phenomenon in either the published [19] or pre-print version [18].

This tokenomics model is hence designed to maintain a self-regulating equilibrium between security, participation, and economic sustainability, see Section 5.1. Obviously, the user willing to participate in the protocol, can buy PALI tokens on any exchange service.

2.6.1 Contracts

In this subsection we describe the contract tokenomics, also showing concrete examples of implementation, which are inspired by possible typical real-world applications of the Palliora framework.

A Palliora contract includes a series of tasks that the contract's actors are supposed to implement, such as: providing data of a certain type or format, providing source code, providing an AI model, performing calculations starting from these, performing verifications, etc.

For example, the contract might require tests on the input and output of each part of the work (see Section 5.1). These tests will be part of the contract and will be implemented by the verifiers, who will then decide to deliver the rewards related to the various steps of the work. A contract may involve several verifiers in a specific test (e.g., the output quality), where passing or failing the tests is decided by the majority of verifiers. In this case, only the verifiers siding with the majority decision will receive their fee for the job done (see Section 5.1).

Contracts will be formalizable and the various options will be selectable through a specific tool developed in Palliora.

We summarize here the content of a typical contract with a particular focus on the parts of the contract concerning the tokenomics. We will then illustrate, using concrete examples, the role of the contract's fields, along with a discussion on how costs and rewards are distributed.

A contract is fully encoded in one (or more) transaction through the (DAA, DAC) logic (Section 2.5). The fields of specific tokenomics interest are:

- lifetime;
- the actors of related transactions: publishers, retrievers, standard guardians, that is, the $\mathcal{P}_i$'s, the $\mathcal{R}_i$'s and the $\mathcal{G}_i$'s;
- links to the input information (possibly the output of another contract, see Section 5.1);
- formal tests $\mathcal{T}_{i:1}, \dots, \mathcal{T}_{i:m}$ to be performed on the *input*;
- fees $\mathcal{F}_{i:1}, \dots, \mathcal{F}_{i:m}$ which will be delivered by the verifiers to the actors upon passing the input tests¹²
- privacy settings for the computation;
- calculators involved $\mathcal{C}_1, \dots, \mathcal{C}_l$;
- tests applied $\mathcal{T}_{o:1}, \dots, \mathcal{T}_{o:k}$ to be performed on the *output*;
- fees $\mathcal{F}_{o:1}, \dots, \mathcal{F}_{o:k}$ proposed for the output of each computation;

¹²A fee may be a vector, specifying the amount of tokens to be delivered to the actors.

- verifiers involved $\mathcal{V}_1, \dots, \mathcal{V}_h$;
- Fees $F_{\mathcal{V}:1}, \dots, F_{\mathcal{V}:(k+m)}$ proposed for each verification.

The first example we describe tackles the training of a model. Suppose there are two users, $\mathcal{R}$ and $\mathcal{P}$, where

- ▷ after developing an AI model, $\mathcal{R}$ wants to train it with more data, but keeping it confidential.
- ▷ $\mathcal{P}$ agrees with $\mathcal{R}$ to provide some data, in a certain format. $\mathcal{P}$ can also have the interest of keeping $\mathcal{P}$'s data confidential. $\mathcal{P}$ will receive a fee F_1 if data (the *input*) pass a quality test T_1 , plus an additional fee F_2 if at the end of the computation the trained model (the *output*) passes a test T_2 ¹³.

This can be formalized in a *contract* implemented in Palliora by the following sequence of steps.

1. $\mathcal{R}$ agrees off-chain (on the marketplace) with $\mathcal{P}$, $\mathcal{C}$, $\mathcal{G}$, $\mathcal{V}$ on the above course of action (with $\mathcal{V}$ checking both input and output). The contract involves $\mathcal{P}, \mathcal{R}$, a calculator $\mathcal{C}$, a guardian $\mathcal{G}$, a verifier $\mathcal{V}$;
2. $\mathcal{R}$ formalizes and uploads the contract on the DA layer ($\mathcal{R}$ pays the due fees to DA). The contract contains a description of the work, including the fees: F_1, F_2 for $\mathcal{P}$, F_3 and F_6 for $\mathcal{C}$, F_4 for $\mathcal{V}$, F_5 for $\mathcal{G}$.

Furthermore, it includes the test T_1 for the input (which releases F_1), and the test T_2 for the output. The latter is composed by two tests: a verification on the quality of the computation done, with associated fee F_3 for $\mathcal{C}$, and a performance test on the final output, with associated fee F_2 for $\mathcal{P}$ and F_6 for $\mathcal{C}$ (if the trained model passes the test on the benchmark). The contract also contains a link to the (encrypted) source code from $\mathcal{R}$ and the (encrypted) data from $\mathcal{P}$ (all information is provided with the due disclosure constraints).

3. $\mathcal{R}$ freezes the total of the promised fees in its wallet, to guarantee the payments.
4. As soon the actors $\mathcal{P}$, $\mathcal{C}$, $\mathcal{G}$, $\mathcal{V}$ sign the contract, the actual work starts.
5. The verifier checks all signatures and executes the test T_1 on the input. In case the test is passed, $\mathcal{V}$ transfers F_1 to $\mathcal{P}$ and further validates the contract. If the test fails, $\mathcal{P}$ is charged of the DA fees and the verifiers fees, the contract is closed.
6. After this validation, the calculator can start the work. $\mathcal{C}$ accesses the code and data and performs the computation work with the due logic. $\mathcal{C}$ uploads the results on the DA, paying the fee.
7. The verifier checks the output by applying T_2 . If the test passes, $\mathcal{V}$ delivers the fees F_6 to $\mathcal{C}$ and F_2 to $\mathcal{P}$.
8. Either the case, $\mathcal{G}$ and $\mathcal{V}$ will obtain F_5 and F_4 if they performed its duties.

Concrete instances of the distribution of rewards among the various actors in this kind of contract are given in Tables 1 and 2. The two cases considered refer to possible different situations, where the value of the data provided, or the cost of the computation prevails. The examples also include a different engagement of the guardians, due to different data privacy settings.

The second example abstracts a data analysis service. Suppose there are two users, $\mathcal{R}$ and $\mathcal{P}$, where:

- ▷ $\mathcal{R}$ has a set of data and they search for a software tool (AI or other) to analyse it.
- ▷ $\mathcal{P}$ possesses a suitable data analysis tool.
- ▷ $\mathcal{R}$ agrees with $\mathcal{P}$ that fee F_1 will be delivered to $\mathcal{P}$ on passing a test T_1 . However, neither $\mathcal{R}$ wants to disclose the data to $\mathcal{P}$, nor $\mathcal{P}$ wants to disclose its tool to $\mathcal{R}$.

¹³For example, if the trained model has a better performance on a agreed benchmark.

Recipient	Contribution	Percentage
<i>Standard Remuneration</i>		
Data Provider, $\mathcal{P}$	providing good quality data for the training	$F_1 = 40\%$
Calculator, $\mathcal{C}$	training and optimizing the model	$F_3 = 20\%$
Guardian(s), $\mathcal{G}$	managing the process with the prescribed privacy constraints	$F_4 = 20\%$
Verifier(s), $\mathcal{V}$	verifying the quality of input and output and supervising the process	$F_5 = 20\%$
<i>Bonus Remuneration</i>		
Data Provider, $\mathcal{P}$	quality of the data led to high performance model	$F_2 = 50\%$
Calculator, $\mathcal{C}$	quality of the training led to high performance model	$F_6 = 50\%$

Table 1: Example of remuneration fees for a service of *model training with valuable data and restrictive privacy constraints*. The percentages represent the fees F_i due to each actor in the contract with respect to the total amount of the contract default remuneration (the budget planned for the work, excluding the bonus for high performance) or of the bonus remuneration (provided in case of high performance).

Recipient	Contribution	Percentage
<i>Standard Remuneration</i>		
Data Provider, $\mathcal{P}$	providing good quality data for the training	$F_1 = 20\%$
Calculator, $\mathcal{C}$	training and optimizing the model	$F_3 = 60\%$
Guardian(s), $\mathcal{G}$	managing the process with the prescribed privacy constraints	$F_4 = 10\%$
Verifier(s), $\mathcal{V}$	verifying the quality of input and output and supervising the process	$F_5 = 10\%$
<i>Bonus Remuneration</i>		
Data Provider, $\mathcal{P}$	quality of the data led to high performance model	$F_2 = 50\%$
Calculator, $\mathcal{C}$	quality of the training led to high performance model	$F_6 = 50\%$

Table 2: Example of remuneration fees for a service of *model training with heavy computational needs and low privacy constraints*. The percentages represent the fees F_i due to each actor in the contract with respect to the total amount of the contract default remuneration (the budget planned for the work, excluding the bonus for high performance) or of the bonus remuneration (provided in case of high performance).

1. The contract involves $\mathcal{R}$, $\mathcal{P}$, calculator $\mathcal{C}$, verifier $\mathcal{V}$ and standard guardian $\mathcal{G}$.
2. $\mathcal{R}$ uploads on the DA layer (pays the fee to DA) the contract to be signed. The contract includes the list of actors, a test T_1 on the output of the computation, as well relative fees F_1 and F_2 to be respectively delivered to $\mathcal{P}$, in case the data analysis is satisfactory, and to $\mathcal{C}$ in case the computation was satisfactory and respected the privacy constraints. The contract contains also fees F_3 , for the verifier, and F_4 , for $\mathcal{G}$.
3. When the contract is signed and validated, the calculator starts the work and uploads the output ($\mathcal{C}$ pays the DA fees).
4. The verifiers execute T_1 .
 - If T_1 passes, F_1 is delivered to $\mathcal{P}$ and $\mathcal{R}$ obtains access to the output.
 - If T_1 fails, F_1 is not delivered to $\mathcal{P}$ and $\mathcal{R}$ does not obtain access to the output. Rather, $\mathcal{P}$ or/and $\mathcal{C}$ is charged of the DA fees paid by $\mathcal{R}$ (according to the contract's details).

The contract is closed.

This kind of contract might include a further test T_2 on the output, to deliver bonus fees F_5, F_6 to $\mathcal{P}$ and $\mathcal{C}$ in case of an excellent performance of the data analysis tool/service as in the previous model training example. An instance of the distribution of rewards among the various actors in the contract is given in Table 3.

Recipient	Contribution	Percentage
<i>Standard Remuneration</i>		
Tool provider, $\mathcal{P}$	providing good quality data analysis	$F_1 = 60\%$
Calculator, $\mathcal{C}$	performing the computation with the prescribed privacy constraints	$F_2 = 20\%$
Guardian(s), $\mathcal{G}$	managing the process with the prescribed privacy constraints	$F_4 = 10\%$
Verifier(s), $\mathcal{V}$	verifying the quality of the output and supervising the process	$F_3 = 10\%$
<i>Bonus Remuneration</i>		
Tool provider, $\mathcal{P}$	the tool was particularly effective	$F_5 = 60\%$
Calculator, $\mathcal{C}$	participating to a successful analysis	$F_6 = 40\%$

Table 3: Example of remuneration fees for the data analysis service contract. The percentage represent the fees F_i due to each actor in the contract with respect to the total amount of the contract default remuneration (the budget planned for the work, excluding the bonus for high performance) or of the bonus remuneration.

2.6.2 The DA Layer

In this subsection we give more details about the system of rewards related to the DA layer (Section 2.2). At the beginning of this section we sketched the staking rewarding, now we detail how the DA fees are determined.

Each transaction incurs a fee, which is mostly delivered to the miners realizing it and the stakers supporting them, see Section 5.1. The total fee charged for a transaction is computed as the sum of multiple components, as described below:

- *minimal fee*: a fee, decided by the Fee Pricing Committee that applies to every transaction, which acts as a rough estimate of the computational cost of the transaction.
- *size fee*: this component is proportional to the size of the data included in the transaction. Larger data loads result in higher size fees.
- *congestion fee*: a dynamic fee component that depends on the relationship between the network/computational resources required to process the transaction and the currently-available resources. This allows the network to adapt fees in response to changing economic conditions.
- *tip (optional)*: a fee that users can include to incentivize miners to prioritize their transactions.

To assist users in estimating transaction fees, Palliora will provide a *Fee Estimation Tool*. Users can input the size of the data they intend to submit and the due confidentiality features to receive an estimated fee for their transaction.

2.6.3 Further Remarks on Rewards

In any application scenario, it is essential to establish a rewarding mechanism. In some cases, it reduces to someone offering services/data and someone else paying an agreed price for using such service/data. In more complex situations, many actors can jointly contribute to create something which is very valuable, and so it becomes difficult to fairly reward each actor. In Palliora we must pay special attention to the evaluation of input and output, as detailed below.

Input Quality Palliora's rewarding mechanisms must take into account the quality of input, not only to filter fake or bad quality submissions. More precisely, we identify two types of data source quality:

- *a posteriori*, Palliora might reward based on an implemented "objective analysis" of the data quality (e.g., evaluating the model performance after the data are used w.r.t. some open metrics). This is very difficult to accomplish and currently an open problem, because the actors require

assurance that the metrics have been fairly applied/computed which can only be done a posteriori, typically after the training set is applied to a set and the model gets fine-tuned. The choice of metrics is therefore paramount to guarantee fairness and its evaluation must be verifiable too. Additional costs are inevitable to effectively verify the data source.

- *a priori*, that is, the data source is trusted. In this case Palliora pays automatically as soon as the data sources is verified to be correctly formed. An example can be a device providing biometric measurements obtained via a TEE-like device (e.g. a smart-watch). The measurements are trusted to be correct, but we need to verify their format. Since we do not need to verify the data source, we incur only in the costs for the data verification.

Output Quality The rewarding mechanism based on the output quality has to be very flexible because of the variety of possible applications. For example:

- the cost of the quality check is not significantly lower than the computation of the output itself;
an example might be to compare a model training that depends on a parameter ρ and the computation goal is to find the best model as a function of ρ ; in this case the only reasonable test could be to train many models according to random values of ρ and compare them on a data set with the calculation output;
- the quality can be directly evaluated by a quick algorithm on the output;
- the quality evaluation requires additional information to be performed;
for example, some certificate/proofs are needed to show the (sub-)optimality of the solution.

Therefore, the output quality can only be ingrained inside the contracts (including tests to be run, similarly to the input) as previously explained.

2.7 Interaction with the Outside: Bridges/Oracles/dApps

Palliora's interaction with the "outside", that is, with services on the Internet and with other blockchains, is facilitated by two core aspects of Palliora's design:

- its DA layer,
- the input/output structure of its transactions,

as we elaborate below.

Palliora allows for natural data exchange with the outside primarily thanks to two facts related to its DA layer. First, Palliora incorporates its own DA layer frictionless by design. Second, Palliora users may leverage this layer (if they wish so) by exploiting only its DA functionalities. For example, a cross-chain bridge may deposit (external) wallet accounts directly in the DA layer, while Palliora apps can retrieve them, act on them and then deposit back the results, letting the cross-chain bridge retrieve the outcome.

As regards the interaction with the outside, another important feature of Palliora is the format of its own transactions, in particular the fields "pointer to locations of input" and "pointer to location of output". These pointers represent a natural contact point with the outside and can be anything, for example, an "input location" can be a call to a web API, or more generally an oracle.

The role of Palliora dApps is worth mentioning. Palliora contracts have a very complex structure, compared with more traditional programmable blockchains. Thus the best way for most users to take advantage of Palliora's features and innovation is via specific dApps. Indeed, only a well-designed dApp can navigate through the complexities of Palliora's network and protocols to maximise its benefits for any specific use case.

In conclusion, the combination of our advanced DA layer and the accurately-designed format allows for external applications (including rollups) to integrate easily with Palliora.

2.8 Palliora Example Functionalities Instantiation

We conclude the technical section by providing two abstraction of realistic examples to highlight how the Palliora's protocol can be effectively instantiated and provide an intuition on the elemental components used as building blocks for the later-presented real applications.

2.8.1 Outsource Computations to Known Actors

The first example considers a publisher $\mathcal{P}$ willing to request the evaluation of f on the data d by a calculator $\mathcal{C}$ of which there is a verification function g that outputs if the evaluation is correct. The input d provided is to be considered valid and $\mathcal{C}$ is trusted to confidentially handle d . The evaluation should be provided to the retriever $\mathcal{R}$ that can access the output only after a day after the contract was signed.

This example abstracts any model evaluation or training process where the verification function can effectively be substituted with an accuracy evaluation or a specific test on a private dataset. Observe that both $\mathcal{C}$ and $\mathcal{R}$ are known in the example representing applications where every AI actor is decided since the beginning. We explain how to deal with unknown actors later on in the second example.

Providing Data to the Calculator. First, $\mathcal{P}$, $\mathcal{C}$ and $\mathcal{R}$ agrees on how the computation must be made, the rewards and the cipher-suite to use. For our example, we assume they agree in having a 3-out-of-5 threshold for the guardians' interactions, use the public-key encryption scheme (Enc, Dec) which allows double decryption for both the retriever and a validator, and agree on requesting the verifier $\mathcal{V}$ to validate the computed output.

All the information are encoded in the (DAA, DAC) logic defining the contract tx_0 , we list the fields relevant to our example:

- *cipher-suite*: (Enc, Dec), secret sharing;
- *Crypto-parameters*: (3, 5)-threshold;
- *guardians*: any guardian;
- *retriever*: $\mathcal{C}$ and then $\mathcal{R}$;
- *when*: one day after the contract is confirmed;
- *confidential level*: 0 no confidentiality;
- *location input*: this transaction;
- *location function*: DA indication where f is stored;
- *location output*: Palliora DA;
- *location output verification*: DA indication where g is stored;
- *validator*: $\mathcal{V}$;
- *reward*: if g passes, $\mathcal{R}$ pays $\mathcal{C}$ a fixed amount of tokens;
- *list of transaction*: rx_1 request of data from $\mathcal{C}$, tx_2 where $\mathcal{C}$ uploads output, rx_2 request of output from $\mathcal{R}$ plus $\mathcal{V}$ confirmation.

After agreeing on the contract, $\mathcal{P}$ provides the contract to the guardians network and collects signature from the guardians willing to participate in the contract's execution by providing the signature $\sigma_i \leftarrow \text{Sign}(\text{sk}_{\mathcal{G}_i}, \text{tx}_0)$ of the contract. When 5 signatures are collected, $\mathcal{P}$ encrypts the data $d \leftarrow \text{Enc}(\text{pk}_{\mathcal{C}}, m)$ for the calculator, splits d into shares $\{d_i\}_i$ that are encrypted for each guardian $c_i \leftarrow \text{Enc}(\text{pk}_{\mathcal{G}_i}, d_i)$. $\mathcal{P}$ completes the first transaction by adding the DA's fees and signing. The whole transaction is provided to the guardians.

The guardians verify the transaction, stores the relevant information for later quick retrieval, sign and push the transaction tx_1 for mining in the DA blockchain.

Retrieving Datum and Uploading Computations To retrieve the input data, $\mathcal{C}$ request by creating a retrieving transaction rx_1 , as indicated in the contract. For our example, rx_1 mainly contains the $\mathcal{C}$ identity and the appropriate identifier to tx_1 , the guardians fees plus the data retrieval from the DA layer. The guardians verify the legitimacy of the request in their stored values and, if valid, request the encrypted shares from the DA layer, decrypts $\text{d}_i = \text{Dec}(\text{sk}_{\mathcal{G}_i}, \text{c}_i)$ and provide the reconstructed datum d in the DA layer for traceability and from which can be retrieved by $\mathcal{C}$. $\mathcal{C}$ decrypts the datum $\text{m} = \text{Dec}(\text{sk}_{\mathcal{C}}, \text{d})$ and executes the agreed calculation $f(\text{m}) = \text{y}$.

Similarly to before, $\mathcal{C}$ creates the transaction tx_2 for uploading the output in the DA layer where the major differences is that the output is prepared by encrypting it for both $\mathcal{R}$ and $\mathcal{V}$. Clearly, the transaction might have different fields but it must point to the original contract agreed in tx_1 .

Validation and Final Retrieval The final retrieval is initiated by $\mathcal{R}$ that creates a request transaction rx_2 indicating the transaction tx_2 containing the output and by providing the agreed rewards to $\mathcal{C}$, plus the additional guardians, verifier and DA's fees. The guardians verify and provide to $\mathcal{V}$ the encrypted output which is decrypted and used to compute the verification function $g(\text{y})$. For our example, the verifier executes $g(\text{y})$ and check that it passes the verification test and, if so, prepares a signed transaction indicating the output passed the verification by providing the reconstructed encrypted output.

The transaction is mined concluding the contract and finalizing the rewarding of $\mathcal{C}$, the guardians, $\mathcal{V}$ and the DA miners.

2.8.2 Dealing with Unknown Actors

The second example describes how a publisher $\mathcal{P}$ uploads data d to the Palliora DA before deciding on a designated retriever $\mathcal{R}$. The data is always confidential, even to the standard guardians handling the contract.

This example is core to any application where an input/output datum is confidentially stored in the DA for future usage. We identify three possible methodologies to encode such requirements in the Palliora contract: the first is if $\mathcal{P}$ is required to further participate in the protocol, the second is if $\mathcal{P}$ is not, the third is a mixed approach. We expect $\mathcal{P}$ to identify $\mathcal{R}$ at a later point in time with respect to the initial data upload. When this happens, $\mathcal{P}$ constructs an authenticated request σ specifying $\mathcal{R}$ as the designated retriever, e.g. $\mathcal{P}$ can compute a signature $\sigma \leftarrow \text{Sign}(\text{sk}_{\mathcal{P}}, \text{pk}_{\mathcal{R}}, \tau)$ of $\mathcal{R}$'s public key together with some additional information τ . The methodologies, some of which are highlighted in Figure 7, are:

- if $\mathcal{P}$ provides the σ to $\mathcal{R}$, then $\mathcal{P}$ is not required to further actively participate in the protocol. To handle this scenario, the DAA must specify a *proxy re-encryption* encryption scheme, where the proxy is handled by the standard guardians¹⁴. $\mathcal{R}$ must provide in its retrieving transaction rx the authenticated request σ . The standard guardians can verify the request and, if correct, execute the re-encryption with the public key $\text{pk}_{\mathcal{R}}$, thus allowing $\mathcal{R}$ to obtain the data.
- if $\mathcal{P}$ does not provide σ to $\mathcal{R}$, then $\mathcal{P}$ must provide σ to the guardians, as well as the secret key sk for decrypting the data to $\mathcal{R}$. This can be accomplished by creating an initial contract that includes additional checks into the field “*additional conditions*” of the DAA_2 (see Section 2.5). More specifically, the check must verify that a second transaction from $\mathcal{P}$ exists and that it contains an authenticated request σ indicating $\text{pk}_{\mathcal{R}}$ as the identity of whoever is requesting the data ($\mathcal{R}$). Until $\mathcal{P}$ issues the second transaction (which happens when $\mathcal{R}$ is identified), the data is securely locked in Palliora's DA. $\mathcal{P}$'s second transaction contains both σ and $\text{Enc}(\text{pk}_{\mathcal{R}}, \text{sk})$.

At this point, $\mathcal{R}$ can request the data, since the guardians have all transactions required to verify that $\mathcal{R}$ is the designated retriever. They will then send to $\mathcal{R}$ both ciphertexts $\text{Enc}(\text{pk}, \text{m})$ and $\text{Enc}(\text{pk}_{\mathcal{R}}, \text{sk})$, from which $\mathcal{R}$ can (first) obtain the decrypting key and (then) the data.

¹⁴Such an encryption scheme allows the standard guardians to transform a ciphertext c encrypted by $\mathcal{P}$ for a public key pk into a different ciphertext $\hat{\text{c}}$ (with same plaintext) for the retriever's public key $\text{pk}_{\mathcal{R}}$.

→ a mixed approach can be followed if $\mathcal{P}$ provides both σ and $\text{Enc}(\text{pk}_{\mathcal{R}}, \text{sk})$ directly to $\mathcal{R}$, when $\mathcal{R}$ is identified. This avoids complex encryption schemes and reduces the number of transactions required for retrieving the data.

Despite the last two methodologies are less computationally expensive, they are risky because we must assume that $\mathcal{P}$ honestly provides the correct encryption $\text{Enc}(\text{pk}_{\mathcal{R}}, \text{sk})$. If fully untrusted computation must be guaranteed, additional proof of correct encryption must be provided (e.g. SNARK/STARK, see Section 2.4.1).

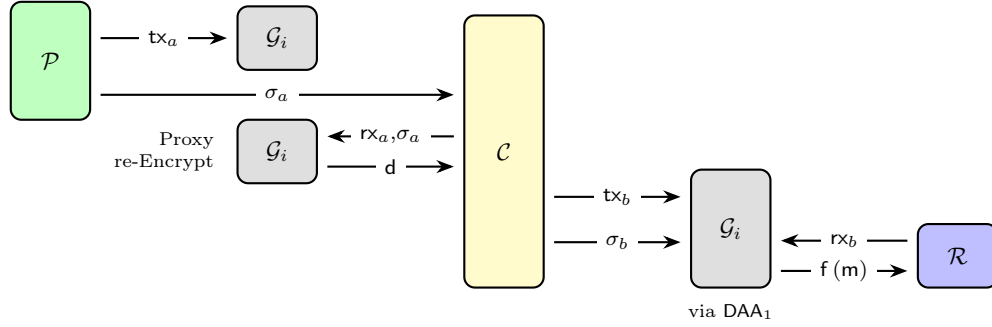


Figure 7: High-level overview of two solutions for the unknown actors. We consider $\mathcal{P}$ and unknown $\mathcal{C}$ and $\mathcal{R}$ for the unknown version of the outsourced computational example of Section 2.8.1) which comprises two exchanges, i.e. $\mathcal{P} \rightarrow \mathcal{C}$ and $\mathcal{C} \rightarrow \mathcal{R}$. We highlight the first two solutions described, i.e. via proxy re-encryption and via second transaction plus DAA_1 additional check.

More complex scenarios are possible (see Figure 7). For example, we can consider the case when both $\mathcal{C}$ and $\mathcal{R}$ are unknown. This case can be solved by adapting the second methodology in the above list. In particular, $\mathcal{P}$ will provide to the guardians σ and the encrypted secret keys for both $\mathcal{C}$ and $\mathcal{R}$.

3 Palliora's Applications

Palliora system can be used in numerous applications where selective data sharing, confidential computing and verification are needed, with rewards and incentives aligned within the participants. In this section we start (Section 3.1) with its application as a general-purpose tool to Finance, replacing existing suboptimal solutions which involve an intermediary that needs to be trusted without verification guarantees. We then (Section 3.2) focus on other blockchain-related scenarios, before discussing in detail (Section 3.3) the applicability of Palliora in AI applications. Finally, we hint at other possible use-cases (Section 3.4), in order to show Palliora versatility.

3.1 Finance

1. **Compliant and private transactions:** with *decentralized compliance* we mean a situation where the published data is shared with a regulator that is allowed to access the data only if certain conditions are met. Consider the following example concerning financial transactions. The transaction's metadata might be encrypted by the application to provide privacy and anonymity by design. However, to guarantee a traceable audit in the event of illicit fund handling, a regulator is allowed to review the encrypted metadata and act accordingly if the transaction is not compliant. More generally, all applications where a trusted entity is required for executing an audit or fraud detection (on the application's data or process execution) can use Palliora to provide a decentralized methodology to allow only designated actors the ability to decrypt and recover private information, but only when certain conditions are met.
2. **Credit assessment while preserving privacy:** a group of lenders and borrowers can share a wide variety of data among themselves and analyse it using strong confidential guarantees provided by the Palliora system. In the current system, the borrower has to trust the intermediary to perform an accurate risk assessment, whereas Palliora can offer verifiable guarantees without exposing information to unauthorized parties. This also enables on-chain credit scoring that can be used to offer loans.
3. **Privacy-preserving market design:** a Decentralized Finance (DeFi) trading dApp can be built on Palliora that can compute asset prices without revealing sensitive information to other traders. Traders may face significant losses because transparent markets expose private information to everyone. This leads them to seek private markets controlled by a single entity, which involve high intermediation costs and the common risk of centralization (misuse, leaks, etc.). On the contrary, Palliora has the inherent capability to design trading engines using dApp logic, verification, and computation without revealing sensitive data.

3.2 Privacy in blockchain execution

Execution of smart contracts is probably one of the most interesting properties of blockchain technology. However, in order to guarantee exact execution, classical smart contract execution faces two challenges: the program is replicated by many nodes (the more, the better), and its inner workings, as well as its input and output, are totally public, leaving no privacy or confidentiality to the user. While there are proposed solutions [16, 23, 87] to these challenges, a unified solution is missing.

In this regard, Palliora can be viewed as a *hybrid* system where data accountability can be used to delink the execution to external parties like *private* rollup sequencers¹⁵ gas while introducing privacy and confidentiality. In other words, users (or dApps) can outsource computations to the guardians' network in a trusted-compute or confidential-compute manner. We list some examples of smart contract computations that should be kept trusted: cryptographic verifications (e.g., zero-knowledge proofs), private smart-contract logic (like parameter optimisation in Decentralized Finance models), FHE-rollups on encrypted data (which need Palliora to manage keys and user access). This may look

¹⁵A *private* rollup relies on a smart-contract sequencer, which is secret but trusted by the users to minimize fairly

similar to the new modular thesis in blockchain, where ledgers do not themselves perform general-purpose blockchain computation but rather leverage on Data Availability to offload the responsibility of smart contract execution to other Layer-2 networks. And yet, Palliora's paradigm natively contains also confidentiality assurance, which is otherwise missing. Therefore, Palliora solves, in a unified solution, both above-mentioned challenges in blockchain technology.

3.3 AI Application

We provide several examples of AI dApps using Palliora as users, of which the most relevant and later discussed in details.

1. **Data Sharing for AI:** a $\mathcal{B}_A$ needs to train its AI model with personal data of thousands/millions of people. However, privacy concerns prevent people from sharing their data, even if $\mathcal{B}_A$ is ready to financially reward the data providers. To overcome this problem, $\mathcal{B}_A$ can set up a Palliora dApp, which collects (in a privacy-preserving way) personal data via interaction with Palliora and pays the data providers automatically, so that their privacy and reward are guaranteed.
2. **Open Source AI:** Open Source AI developers $\mathcal{D}_C$ improve on each others' models, until one model reaches a quality target (possibly w.r.t. some open metrics) that makes the fine-tuned model valuable for end users. A Palliora dApp can keep track of $\mathcal{D}_C$'s work (guaranteeing integrity and authenticity) and financially reward all the contributors to the final model.
3. **Encrypted Models for AI:** a business $\mathcal{B}_B$ possesses a large amount of data related to its business. $\mathcal{B}_B$ wants to find an AI model that best fits its dataset and that provides accurate predictions for its market. However, some model developers consider their models an industrial secret, so they are unwilling to share the models with $\mathcal{B}_B$. A Palliora dApp trusted by $\mathcal{B}_B$ can interact with model developers and (confidentially) test their models with $\mathcal{B}_B$'s dataset, without having access to their model in cleartext. The selected model will be notified, and then $\mathcal{B}_B$ and its developers can negotiate its use.
4. **Decentralized Private AI Cloud:** a dApp $\mathcal{A}_E$ can act as a front-end for the decentralisation analogue of the newly-formed centralised Artificial Intelligence system (e.g., [3]), simultaneously providing multiple trusted computation parties that offer AI capabilities. This way, $\mathcal{A}_E$ can give people (that want to locally train an AI model on their personal data) privacy guarantees to choose their computation provider, while preserving their data privacy.
5. **AI Marketplace:** a network of Palliora dApps could act as a huge *marketplace* where all AI actors interact and make deals. For example, data owners might sell their data according to demand, selectively and in a controlled way. The same could also be done by model developers. Even more advanced applications can be built on top of this dApps network, such as dApps that allow end users $\mathcal{U}_D$ to ask for brand-new models specific to their needs, while the AI actors make deals on how to reach $\mathcal{U}_D$'s goal and how to distribute its reward.
6. **Verifiable and Decentralized AI Agents:** autonomous AI agents lack trust because end users need them to handle personal data or preferences and want them to be accountable for their decisions. Agent developers can deploy agents that remain private and verifiable while using users' personal data to act according to their predefined logic. The computation of such agents can be further performed using the Palliora's guardian nodes. Additionally, AI agents can utilise the already running Decentralized Private AI Cloud to ensure various levels of confidentiality. Since AI agents are containerized extensions of models and computation workflows, the Palliora's fee and reward systems can be utilized to coordinate them in a dApp-defined marketplace.

3.4 Other Applications

We also provide some more use-cases that show how it is easy to leverage Palliora's selective disclosure of secrets to reach complex goals.

1. **Digital Sovereignty:** Palliora applications can be utilised by the public sector, as confidential computing ensures data remains encrypted during processing, enabling the secure sharing of citizens' data across public environments. Moreover, Palliora supports digital sovereignty by providing fine-grained access control and protection.
2. **Digital Platforms Advertising:** Digital advertising is shifting from exploitation of third-party cookies to more privacy-safe options, possibly restricting cross-site and app tracking. Palliora is an ideal environment for secure user data processing by advertisers, since Palliora's confidential computing techniques can be used in matching algorithms, attribution, and data aggregation.
3. **Health Care and Life sciences (Sharing of personal health data):** With rising healthcare technology users can share their personal data from wearable devices, diagnostic tests and mobile apps for early disease detection, personal recommendation and early alerts. Palliora can then be used by pharmaceutical companies to share securely proprietary datasets, thereby speeding up drug discovery, while enabling collaboration and protecting intellectual property.

4 Details of AI Applications

4.1 Open Source AI

Open-source AI has a dual incentive problem. On one hand, AI model users are concerned about their privacy, so they are reluctant to share their data without either clear incentives or assurance of data confidentiality. On the other hand, AI model developers are not encouraged to put effort into their models when anyone can use them for free [37, 43, 44, 93]. As a consequence, free-and-open AI lags behind AI run by big corporations.

We can consider two examples, as follows. First, where a company needs to train its AI model with personal data but privacy concerns prevent people from sharing their data. Second, where a developer team publicly train an AI model with each improvement traced and properly rewarded.

4.1.1 Relation with the Framework

The data/model development and ownership must be traced during the development of an open AI model. Publishers can upload data in a confidential way, allowing only the intended model developer to decrypt and retrieve the training data. Since everyone can see this model (data immutability and availability required), the contribution of each participant can be tested on an agreed (not tamperable) test set. Only who indeed improves the model performance, either with new training data or new topology, can be rewarded.

The tracing contributions problem can be solved by setting up a dApp that collects Palliora contracts that identify all the data/code improvements and, once some verifiable goal is reached, the rewards are automatically provided. The dApp keeps track of developers work (integrity and authenticity) and rewards all contributors to the final model as described by the DAA/DAC logic in the contract. It is also essential to keep track of the model ownership which can easily be done, thanks to the blockchain properties of the framework.

After the model is declared good, i.e. the model has reached an accuracy target suggesting the boost in its economic value, all potential buyers can test the model and pay when they are happy to get the model thus rewarding accordingly all the actors involved during the model training. To ensure payment and avoid the direct usage of the open model, we might implement several strategies, for example time-locking.

Regarding the time-locking solution for a controlled test of the model without full reveal, we consider that the model's performance is published but not the model itself, so that the buyer interested in using the model will pay first and later have access to the model with the guarantee that their money will

be returned in case the claimed performance are not achieved. The Palliora framework can guarantee both time-locking (via DAA logic) and a posteriori check on the performance (via DAC logic).

4.1.2 Palliora’s Applicability

Generally speaking, a dApp can use Palliora as a base layer to incentivise users to share their data for the fine-tuning or training of AI models. The dApp will act as a publisher within Palliora.

As far as model developers are concerned, a Palliora dApp can be used to keep track of developers work to a model improvement. First of all, Palliora guarantees integrity and authenticity of the work done on AI models. Moreover, since the framework keeps track of every intervention, any contributor to the final model can be rewarded. The work on a model can be considered done when it is valuable for some users, so if it reaches a quality target (possibly with respect to some metrics).

In practice, the initial model will probably be a public-foundation model like LAMMA2 [88], to accelerate the initial development and to keep costs low. The users will interact with the dApp (which will probably be managed by either a group of AI developers or it could even be a *data marketplace*, see Section 4.3) and they will be rewarded for providing private annotations to train the initial model. Retrievers can fetch the fine-tuned version of the model when it has reached a quality target (e.g., according to some open metrics) that would make it significantly more valuable. If they are happy with the new model, they will pay the entities that funded the dApp. Then, the retrievers will have private access to a private model, but only for a limited period. After this phase and another possible access embargo, the guardians will open up access to everyone. This is possible thanks to Palliora’s time-locking properties, highly flexible policies (selected access and trusted/confidential computation), and the underlying high confidentiality provided by cryptography.

For these reasons, Palliora can solve the dual incentive problem. Data providers are rewarded to train the models and models developers have their work repaid, and each developer can work on the improvement of any open AI model.

4.2 Encrypted Models for AI

When considering AI application, one question that comes to mind is how to provide additional confidentiality guarantees to further achieve higher level of privacy on both data and/or the computation executed.

Let us explain the problem with a fictitious example. Consider the task of identify if a given hand-written signature is produced by a left-handed writer, for which many companies claim to have a very good model, e.g. a classifier. A company wants to use the best model to solve the task, where “*best*” relates to the performance on some (private) data belonging to the company, implying a non-public test-set. The company is ready to pay a suitable amount and not only the data owned by the company are secret, but even the models are secret, and their owners are unwilling to share them without guarantees.

Other real examples [45, 69, 84] might include similar privacy-oriented application, e.g. executing grammar correction without revealing the text, photo and video editing on private media.

4.2.1 Relation with the Framework

Additional confidentiality guarantees can be reached by considering ad-hoc solutions merging advanced cryptography primitives with AI model executions, e.g. by confidentially training an AI model over encrypted data using FHE, FE or SMPC.

The setup of such confidential computational procedures must be properly executed to avoid involuntary leaks caused by primitive misused, as better explained in Section 2.4.1. Broadly speaking, the actors must agree (during the initial setup procedure) on the cryptographical solution, e.g. an FHE primitive. Such a choice requires all I/O data, model design and evaluation to be compatible with the selected primitive. For the FHE example, in order to allow the calculator $\mathcal{C}$ to evaluate the model homomorphically, all input data must be encrypted and an evaluation key must be given to $\mathcal{C}$.

4.2.2 Palliora’s Applicability

Palliora allows the model developers to confidentially test their secret models with company’s private data as input, that is, without a cleartext access to either models, inputs and/or computations. For our classifier example, each model developer can provide to a calculator an encrypted version of their model which can be homomorphically evaluated by $\mathcal{C}$ and, with the result naturally encrypted. All the encrypted results can be provided to several guardians that execute a secure multi-party protocol to decide which model is the best choice for the company. This solution can be coordinated and described in a Palliora contract.

Additionally, developers who want to keep their model secret can use a Palliora dApp to share them in an encrypted way. Since Palliora’s contracts provide DAC logic, computations can be executed even if the models are encrypted. This means that a Palliora dApp can confidentially test a model. When a company wants to find an AI model that best fits its dataset and provides accurate predictions for its business, a Palliora dApp can test different models, find the best one and communicate it to the company. So, the company and the model providers can negotiate and make some deals. All this work is done without the leakage of sensitive information about the AI models considered.

4.3 AI Marketplace

One of our AI oriented application for Palliora is an *AI marketplace* as a decentralized platform where all the Palliora actors coordinate their requests to sell/buy data, AI models and related services. Generally speaking, it is not obvious how AI actors (data owners, model developers and model trainers) may get financial reward, especially since they are not coordinating their effort (mostly for lack of mutual trust). Let us consider some explanatory examples:

- data owners might want to sell their data according to demand and in a controlled, selected way;
- similarly, model developers might want to sell their model access according to demand, again without losing control on them;
- or even more complicated scenarios, where end-users might request a brand-new model specific to their needs and AI actors can make a deal to produce the model, facilitating an agreement between users and AI actors by suggesting a fair reward distribution.

An AI marketplace, where they can interact and negotiate, is an ideal platform for the making of these deals.

4.3.1 Palliora’s Applicability

Clearly, the AI marketplace is a very ambitious goal with an intrinsic high complexity, but it is achievable in Palliora.

This application is the natural collection of all the other AI applications described as different Palliora dApps. The marketplace can define Palliora contracts templates that interact with the appropriate dApps to achieve the goal.

Some examples are provided below:

- data owners can see which data are in higher demand (geolocalization, travel habits, sport habits, ...) in order to sell selectively and in a controlled way those particular data.
- models providers can find out the more demanded AI models in order to focus on their improvement and negotiate their sale.
- users can also use the dApps to make specific request. For example, a user can ask for a linear regression model, while a model developer can ask for food habits of people living in Washington.

This marketplace will help the creation and the negotiation of innovative models suiting users’ needs.

4.4 Verifiable and Decentralized AI Agents

One of the prominent application for Palliora is an *AI Agents*. Below highlights the challenges and limitations of current AI Agents, and we explain how Palliora can solve these concerns.

- **Accountability problem in AI agents:** when humans make mistakes, legal, contractual, and ethical systems assign blame or seek compensation. Such structures do not yet exist for AI agents. Currently, the user bears responsibility for errors, with providers denying liability and warning that systems “can make mistakes”. For AI agents to truly perform the capability of high-value tasks, accountability, insurance or warranties are required. However, companies lack incentives to adopt these voluntarily, as building trust is costly while denial remains cheap. Until accountability is established, users bear all risks, likely hindering AI adoption in high-risk and high-value areas.
- **Verifiability problem in AI agents:** AI agents depend on us giving up some control. It’s not just what AI can do, but how much we allow it to decide. Since agency and authority are limited, designing AI involves balancing control among users, programmers, and the system itself, depending on the situation. The main challenge is that AI systems lack built-in ways to verify they act correctly or honestly. As these systems become more complex, this verification gap creates more risks. Trusting autonomous AI while ceding control is tricky—verification is essential to manage this.
- **Centralized ownership problem:** Most AI models are proprietary and mainly centrally controlled. This dependence leads to issues with access, pricing, and updates, which can change unexpectedly, causing agents to malfunction or behave differently. It also means relying heavily on a centralised computing structure that must be trusted for privacy concerns. Open-source models like LLaMA, DeepSeek, and Mistral offer alternatives, but access to large datasets is limited, and scaling requires substantial resources and expertise. Copyright issues related to training data remain unresolved. Relying on closed systems exposes vulnerabilities, but operating private infrastructure is costly and complex.
- **Data access problem:** AI agents will need more personal data to reach full potential, but companies prefer to keep customer data private to prevent users from leaving. Sharing data with third parties is unlikely, risking monopoly issues, so fair competition of data becomes an issue. Users may want more control over their data as the specific agent they only trust has access to data in a secure way with access controls defined for the benefit of the user.

4.4.1 Palliora’s Applicability

Palliora solves the above problem of AI agents by following specific functionalities:

- **Accountability:** the DAC layer can be used to provide accountability guarantees when control over agents resides with a set of guardians. AI agent owners and users can choose which guardians to trust for execution and for disclosing confidential user data with AI agents. This is similar to arbitrary computation accountability, where the AI agent owner’s identity and guardian identities are known, and a set of guardian contracts can include predefined conditions for economic penalties in cases of incorrect execution.
- **Verification:** the verification problem is integrated into the system with some guardians acting as verifiers in the Palliora system. Every step of AI agent outputs can be verified. For example, proof of execution generated by the verifiers can be provided, and only when its correctness is confirmed will the computation or agent decisions proceed.
- **Centralized ownership:** Palliora offers an AI execution cloud that is not reliant on a single entity. The guardians’ network of calculators consists of decentralized compute providers who not only offer verification guarantees but also enable confidential computation, making the AI agents’ execution within the DAC framework both secure and decentralised.
- **Data access problem:** one of the applications we discussed earlier involves sharing confidential data for AI applications. AI agents can use this functionality so that users’ personal data is

shared as intended, with reward mechanisms in place for data sharing. This distinct feature of Palliora provides agents with a data encrypted transmission protocol featuring accountability (DAA) and verified computation (DAC), allowing sovereign AI agents to make decisions on users' behalf with appropriate controls.

4.5 Related Works

Singh et al. [82] provide a complete analysis on the important questions related to decentralize the AI infrastructure to allow a more democratic and fair repartition of rewards for data publishers, model calculator and model providers. Their work develops a solid taxonomy of the different existing solutions in the decentralization spectrum, from centralized AI framework to distributed computing and more. They provide clarifying examples where a decentralized AI infrastructure would be an optimal solution and highlights risks and opportunities for improvements or future research in the domain.

Cheng et al. [27] study the decentralization and democratization of open source AI applications and provide a thorough theoretical classification and requirements for achieving such a solution with a major focus on pros/cons of the different security mechanisms available that guarantee privacy and confidentiality on the data used for model training, the computed models or the models' output. Their proposed model differentiate from Palliora in the fact that their model is designed for the specific open source AI application while Palliora has a more general structure. Such a difference can provide a simplified procedure for, e.g., the fair rewarding of the developer involved in the creation of an AI model while disallow more general contracts, key feature provided by Palliora's DAA/DAC style contract.

We identify a list of public projects that combine AI with the blockchain with the goal of providing either a distributed AI evaluation framework or allow AI agent execution over the blockchain. These projects are in different development phases and (sometime) tackle the decentralized AI problem differently from Palliora. Notably, we identify three complementary project's typology:

- *Ethereum Virtual Machine (EVM) Compatible*: these projects aim at instantiating an AI framework by providing some sort of dApp/smart contract thus allowing a full easy integration with existing EVM [70, 76]. Notably, these solutions have different assumptions and guarantees regarding data confidentiality and accountability because focused on providing direct outsourced-computation services. For this reason, often the AI marketplace does not provide full data control because it works using NFT/token to identify the data input/output [78].

Palliora can interoperate with all these solutions. This would allow Palliora to provide full data accountability while outsourcing the computation to the desired solution.

- *AI-specific Protocols*: these projects aim at providing ad-hoc protocol to execute a single part of the AI data/model flow by augmenting them with additional properties, i.e. an ad-hoc protocol to securely train a model in a decentralized way [42, 68], confidentially execute the model inference or an AI agent [66, 80] or provide an advanced data handling framework [89].

These protocols can be used together with Palliora, providing advanced features and expanding the possible use cases.

- *Research-Driven Solutions*: the last typology we identify contains projects focused on providing generic multi-party computation protocol that allows generic secure computational outsourcing [4, 34, 67, 72]. These projects are mainly research-driven and designed for general usage. All AI applications (like the one described in Section 3) might be possible to achieve with the appropriate considerations.

Palliora might interoperate with such solutions by considering an appropriate separation of responsibilities and duties between the frameworks.

5 Remarks and Comparisons

The last part of the white-paper focuses on highlighting Palliora’s intrinsic limitations and open problems, relevant remark on the protocol’s design and usage, discuss available implementation.

5.1 Remarks

In this section, we collect some important remarks, divided into three subsections:

- remarks on Palliora design, where we explain some design choices and specific properties;
- remarks on Palliora security, where we do not provide formal security proofs, rather we explain the problems, how we tackle them and why our mitigations are successful;
- open problems, that is, challenges that could be solved with further research and that would lead to significant improvements of Palliora.

5.1.1 Design Remarks

We now list seven remarks, four explaining our design choices and three on the initial roll-down of Palliora. Our design choices remarks are:

▷ Honest/quick verifiers and accurate/fast tests

Verifiers and their tests play a crucial role, both for inputs and outputs. Indeed, the verifier(s) $\mathcal{V}$ is the sole responsible for deciding if the input (or the output) is correct and valuable. If $\mathcal{V}$ claims that the test’s result is negative, then the relevant parties (calculator/input providers/etc) will be not rewarded even if the test is inaccurate or $\mathcal{V}$ is reading incorrectly its result. Moreover, $\mathcal{V}$ could be lazy and providing its decision with a huge delay, so that the rewarding is actually blocked. Similarly, if the tests are not accurate, the relevant parties could miss on their rightful reward, which may happen even if they are accurate but extremely slow.

In order to mitigate the occurrence of such unpleasant situations, we designed the contract’s initial setup procedure in such a way that all actors must agree on the actors themselves (without all actors’ signatures, the contract is not valid), their role *and especially* the input/output tests.

▷ Special features of Palliora contracts

Palliora contracts are a special type of computable contracts, described in [17, 19], which differ significantly from usual contracts that can be found in public blockchains (such the standard smart contracts in Ethereum) in at least two significant aspects:

- ▶ Palliora contracts allow for the execution of arbitrary code on arbitrary input, without any constraints on the programming language/environment. Differently from blockchain platforms using general-purpose execution environments¹⁶, Palliora contracts rely on human-readable descriptions, rather than any form of script: they must be proposed by humans and agreed upon by humans¹⁷. This freedom in the contract formulation allows for an extreme flexibility for Palliora tasks, which is of paramount importance in the rapidly-evolving scenario of AI research. Moreover, one of the inputs of a contract may be the output of a previously-executed contract, possibly implementing very complex collaborative work.
- ▶ Because of the extreme flexibility of Palliora’s calculations, Palliora’s contracts do not adopt a gas-based payment system (e.g. as done by Ethereum), rather they leave the determination of the contract’s fees to the free negotiation among parties. This is especially useful for contracts aiming at training a model, which can be a hugely-expensive computational task. Thus, the fee for the computation cannot be determined with simple formulas, but depends on the calculator capabilities. Similarly, the training effort is expected in some cases to be

¹⁶For example, blockchains using web assembly (WASM) like Polkadot [73].

¹⁷When we say "humans" we also include AI agents, or other programmes, that can read and process instructions in natural-language form structure. In other words, we only affirm that the contracts are easy to understand by humans. If humans trust a software to read contract and accept them, this is their decision.

much higher than that for the verifiers, and the relationship between the corresponding fees cannot be standardized in all situations.

▷ **Minimal stakes**

Any Palliora actor is required to stake a minimal amount of tokens to participate in the protocol, but it is very difficult to determine the exact quantities.

Even if we do not detail in this paper these values (which will be decided by us in the first implementation and then adapted by Palliora decentralized governance), we have nonetheless adopted the following principle: for each role, Palliora expects a commitment from the actor to the framework *proportional to the trust* required for that role. In particular, the minimal stake will differ for a calculator, guardian or a verifier.

▷ **Self-regulating tokenomics**

Palliora tokenomics model must maintain a self-regulating equilibrium between security, participation, and economic sustainability.

We now explain how we obtain this balance. First, by tying inflation to the staking rate, the system ensures that staking remains attractive when participation is low, thus reinforcing network security, while naturally tapering inflation as the network becomes more saturated with staked capital. Additionally, by integrating miners cost benchmarks into fee modelling (see Section 2.6.2) and channelling a dynamic share of fees to stakers, the protocol ensures that rewards remain economically grounded.

This layered incentive structure supports a sustainable ecosystem where all actors in the contracts are properly compensated¹⁸, the inflation remains largely predictable, and token holders are consistently encouraged to contribute to the resilience of the network.

The following remarks outline some temporary choices meant to facilitate the growth of Palliora into a sustainable framework.

▷ **DA layer evolution**

The current DA layer based on Substrate is adapted to let Palliora work, bootstrapping our framework. The DA layer is getting refined into an ad-hoc efficient protocol that suits Palliora and its demand, especially in differentiating better the role of DA validators and guardians, enforcing a healthy balance between their competition and their collaboration.

▷ **Multi-role actors**

It is unrealistic to think that Palliora will start with huge activity from the very beginning. Therefore, at the very beginning we must let the same party assume more than one role (like a guardian acting also a DA miner). However, as soon as the network activity grows significantly, we will introduce disincentives that will push parties to specialize in one role, to avoid collusions (which may be potentially lower Palliora security, e.g. if a calculator plays also the role of its own output verifier).

▷ **Tokenomics bootstrap**

To facilitate the bootstrapping of Palliora, all guardians will receive a (significant) inflationary percentage of their stakes. In this way, we will promote the guardians' active participation in Palliora from the beginning, because computing hardware is currently very expensive, especially for massive AI training, and we need to incentivize the owners of these servers to join Palliora. However, the long term design is to progressively reduce such an inflationary additional gain, until the guardians' rewarding source is mainly constituted by service fees.

5.1.2 Security Remarks

In this subsection we explain our choice of design approach "Security & Privacy by Design". To validate the robustness of our design, we also provide several more specific remarks on mitigations of possible attacks/weaknesses.

¹⁸Including verifiers.

▷ Security & Privacy by Design

Palliora’s complex architecture, which consists of a network of DA nodes, a network of (standard) guardians, a subnetwork of calculators and a subnetwork of verifiers, is necessary to provide strong guarantees on the security, privacy and confidentiality in the AI domain (data handling and computations).

To be more precise, note that missing:

- ▶ the DA network, the actors will not be able to securely store and retrieve their data (input, output, model weights, etc.);
- ▶ the (standard) guardian network (working alongside other networks), there will be no enforcement on the correct behaviour by the other actors, there will be no (decentralized) access control w.r.t. data handled by the DA layer;
- ▶ the calculator network (working alongside the standard guardian network), there will be no guarantee on the confidentiality, privacy and correctness of the required computation;
- ▶ the verifiers’ network, there will be no assurance on the quality of data, whether input or output (and on related computations).

▷ Malicious verifiers

If a malicious $\mathcal{V}$ falsely claims that the test’s result is negative, then the relevant parties (calculator/input providers/etc) will be not rewarded. On the other hand, an unscrupulous verifier that wants to minimize its computational work could always declare the test positive, without even running it. Other, more complex unwanted situations might arise, e.g. if a calculator bribes an output verifier.

To prevent these cases, in our design we have introduced three features. First, it is the contract’s creator $\mathcal{P}$ that selects the verifiers, so $\mathcal{P}$ can be helped in its choice by their past behaviour (as in a reputation-based system, which could lead to¹⁹isolate $\mathcal{V}$). Second, the contract may require a proof for the verifiers’ honest behaviour, e.g. by forcing the usage of secure cryptographic primitives. A simple case is when the proof is given by the agreement of a verifiers’ majority (or more complex quorum situations) when many verifiers are involved. A more elaborated case is when the contract requires a (correct test-evaluation) formal proof (e.g. via SNARK/STARK). Finally, it is also possible that the contract forces the slashing of the malicious $\mathcal{V}$ ’s invested stack, this way putting even more pressure on $\mathcal{V}$ to behave correctly.

▷ Malicious test provider

The precise content of each test must be disclosed to each verifier involved, so that they can assess their computational effort and consequently estimate a satisfactory fee. Thus, verifiers who have received the trust of all parties in the contract may *also* verify the fairness of the tests they have to apply. This discourages the test provider from possible fraudulent behaviour (for example, a quality test, which does not verify exactly what was claimed or it is too difficult to pass). In case a verifier deems the test to be unfair or not correct (the verifier must declare its findings on the test), the test’s result will be “pass” by default.

▷ Unknown tests

While tests are always known in advance to verifiers, it may be convenient to keep them unknown to other parties. For example, if the calculator $\mathcal{C}$ knows the output test T , in some situations $\mathcal{C}$ could cheat and present a fake output that satisfies T . (such as when a classifier is tested against a fixed sample set). Similarly, the input provider $\mathcal{P}$ could cheat to satisfy the input test or even provide random data if $\mathcal{P}$ sees that its input would pass it (e.g., when the test merely checks a fixed data format).

As a solution to these problems, in our design we allow for tests on the quality of the input or the output, which are *unknown* to the input provider and/or the calculator. It is up to them to trust the contract creator and sign the contract.

¹⁹All the actors will avoid such verifier, preventing any further reward.

5.1.3 Open Problems

Palliora is a complex systems, with many overlapping networks and internal functionalities/services, therefore there are many areas where it is possible to see paths to significant improvements of Palliora, with further research. In this subsection we highlight some of these open challenges.

▷ Advanced Cryptography

In this paper (the first version of Palliora white paper) we have designed Palliora with the best available cryptographic methods in mind. However, we face two problems that should be addressed in later versions. First, we have never mentioned quantum adversaries and so we have not focussed on quantum-safe cryptography, even if it is a topic which is becoming more and more important. Second, for some parts of our protocols we must necessarily use non-standardized algorithm, for example for the SMPC part where guardians perform threshold cryptography. In both cases, we probably need to wait for the conclusion of current standardization efforts [64, 65] before we can take a decision on the most suitable algorithms/protocols for Palliora.

▷ Tokenomics

A complex system like Palliora has many actors with clearly-distinct roles and it is hard to think of a system where a *totally* fair redistribution of value can happen. While we leave much to the free negotiations among parties, we need a very active platform with thousands of users if you want to reach a balance in the economical scenario. Currently, there are clear incentives for the active participation of DA miners and guardians, generally speaking, but we need also involvement in the staking. To achieve this, in future version, we might tweak the tokenomics somewhat, like for example including a dependence of the inflationary reward also on the guardian share of the total staking.

▷ Other Actors Rewarding

Apart from the main actors described in this white paper, a refined system like Palliora needs other actors, playing a minor role. The main example is given by the so-called *light nodes*, which are present in most blockchain platform and which basically relay information on the block status to the network without receiving a specific reward. Another more peculiar example could be, in a fully-fledged Palliora, a system of DA safety control. Thus, a challenge that newer versions will face will be to identify precisely these minor actors *and* to find a way to compensate them without modifying drastically the current system.

5.2 Available Implementations

We have implemented a working prototype or testnet for Palliora, that we call Manas. On the infrastructure side, Manas consists of a set of *Validators* under Proof of Stake consensus mechanism and *Guardian Network* (that can scale to a few hundred participants). As regards users, Manas enables two crucial roles of *Publishers* and *Retrievers*, who can submit encrypted data (new or duplicate) and access existing data, respectively.

With reference to Section 2.1, Manas Validators maintain the encrypted Data Availability (DA) capabilities of the network, including the storage, verification. Validators maintain the blockchain state, therefore complying with both block production and transaction validation and interactions with Guardians.

On the other hand, Manas Guardians are responsible for enabling both data accountability and confidential computation. The standard Guardian network implements flexible threshold setting among a selected subgroup of Guardian nodes participating in managing access to the data on-chain defined in Section 2.4.1, catering to user-driven/chosen trusted security circle on the Manas network. The Guardian protocol implements the Silent-Threshold Encryption [40] (mainly for its speed and dynamic threshold) as the cryptographic protocol that provides a safe environment for on-chain requests when they are in-transit or at rest in the DA service.

The Guardian network participation is open for anyone who follows the protocol, that is, if they setup a Palliora node client and stake a minimum amount of PALI tokens. During a node client setup, the

operator manages a key-pair (sk, pk) secret to have ownership of rewards and stakes. When the guardian set evolves, then the guardian set parameter gets updated in the system. This update is linked to a configured numbers of blocks (an *epoch*). The key-pair acts as both an identifier (for the node client) and a security parameter for the threshold decryption. Guardian nodes are capable of downloading the computation program in the containerized framework (e.g. Docker) so that Calculators can execute a (trusted or confidential, e.g. TEE) setup. The Manas architecture is shown in Figure 8.

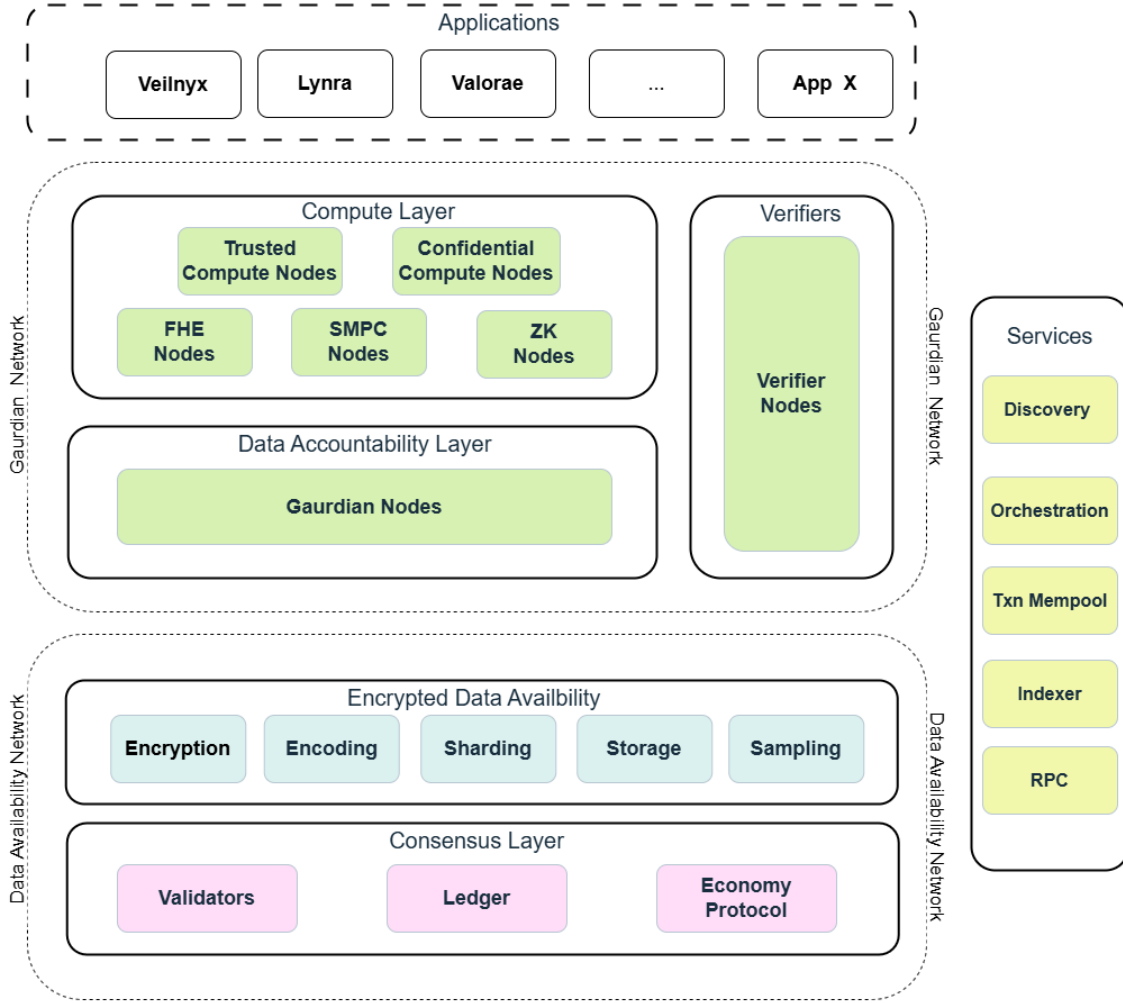


Figure 8: Palliora (Manas testnet) Architecture

Lifecycle of the Request. A *dApp* is a decentralized application that works with a user-owned software wallet. Any dApp interacts with the Manas network through RPC calls, mainly to submit compute requests to the Guardians.

We tested Manas testnet using reference dApps that have been the early users of Palliora:

- ▶ Veilnyx - Compliant privacy by selective disclosures, <https://veilnyx.com/>
- ▶ Valora - Private AI chatbot, <https://valora.tech/>
- ▶ Lynra - Accountable and Private DeFi & AI agents, <https://www.lynra.ai/>
- ▶ Datari - Fine-tuning of LLM models using privately-shared data <https://datari.app/>

The request starts with the data submission in encrypted format, where standard Guardian nodes are entrusted with safeguarding the private data. The Manas Data Availability layer utilizes erasure encoding and KZG commitments [53] to offer recoverability, proof of existence, and a confidence metric to its application layers²⁰. Erasure encoding introduces data redundancy, safeguarding against potential losses with a degree of recoverability. KZG commitments enable efficient proof-of-data existence without revealing the data itself. Once this Guardian receives the compute request, it generates a shared key and decrypts the encrypted request. The decrypted ciphertext contains the necessary payload for the compute service²¹. If the node that receives the request does not process it, even a corrupt node will fail to access the requested data due to the encryption encapsulating it. We follow the protocol specification in Section 2.1 regarding the data sharing mechanism for selective data sharing and price between Publisher and Retriever.

The request flow continues with the dApp creating a Compute Contract Agreement Request. This contract specifies the economic parameters of the computation, including the cost of the computation the user is willing to pay, threshold decryption fees, and the list of Guardians that the user wants to involve. This contract proposal (agreement request) formalizes the terms under which the computation will be executed.

The dApp sends this proposal to an RPC node. The transaction enters the transaction mempool and is broadcast to the specified Guardian nodes. Each Guardian reviews the contract parameters and provides its agreement. Once all required Guardians have signed their agreement on the contract, the finalized contract is included in a block on the Manas blockchain and assigned a unique Contract ID (CID). This CID serves as the reference on the chain for subsequent computation operations. The fees are locked at this point.

After the agreement is finalized on-chain, the dApp submits the actual Compute Request, explicitly referencing the CID. The compute payload is submitted in encrypted format, where the selected Guardian group is entrusted with safeguarding and threshold-decrypting the private data in accordance with the contract.

After decryption, the compute request is forwarded to the designated Computing node (the so-called "Calculators"). As a matter of fact depending on the configuration, in Manas, any Guardian node can be a Standard node or a Computing node or even a Verifier node. As regards Computing nodes, they can be either "trusted" or "confidential", where "trusted" means they can see the input and confidential means they cannot. Users can select the type of Computing nodes. Interestingly, in our current implementation a Manas Guardian can choose to act simultaneously as both a "Trusted" and a "Confidential Compute" node, appropriately serving each type of compute request.

The properties of the Manas network ensure that once decrypted and forwarded according to the contract, the data is processed by the requested computation (e.g., the pre-trained AI model), which generates a suitable response.

The dApp continuously monitors block-production events on the Manas blockchain. Upon detecting the desired result, the dApp fetches it, decrypts it (if necessary at the application layer), and finally the dApp presents it to the user.

Another important aspect of the computation handling is the interaction between blockchain blocks and the dApp. Due to the off-chain nature of the processing, when a Compute request referencing a specific CID reaches the Manas blockchain in block number t , the computation result will not be available until a future block with block number $(t+n)$, where n represents the total time required for agreement finalization, threshold decryption, and computation.

²⁰Thereby facilitating an efficient and reliable system.

²¹The user and dApp ensure the payload is prepared according to the compute service requirements.

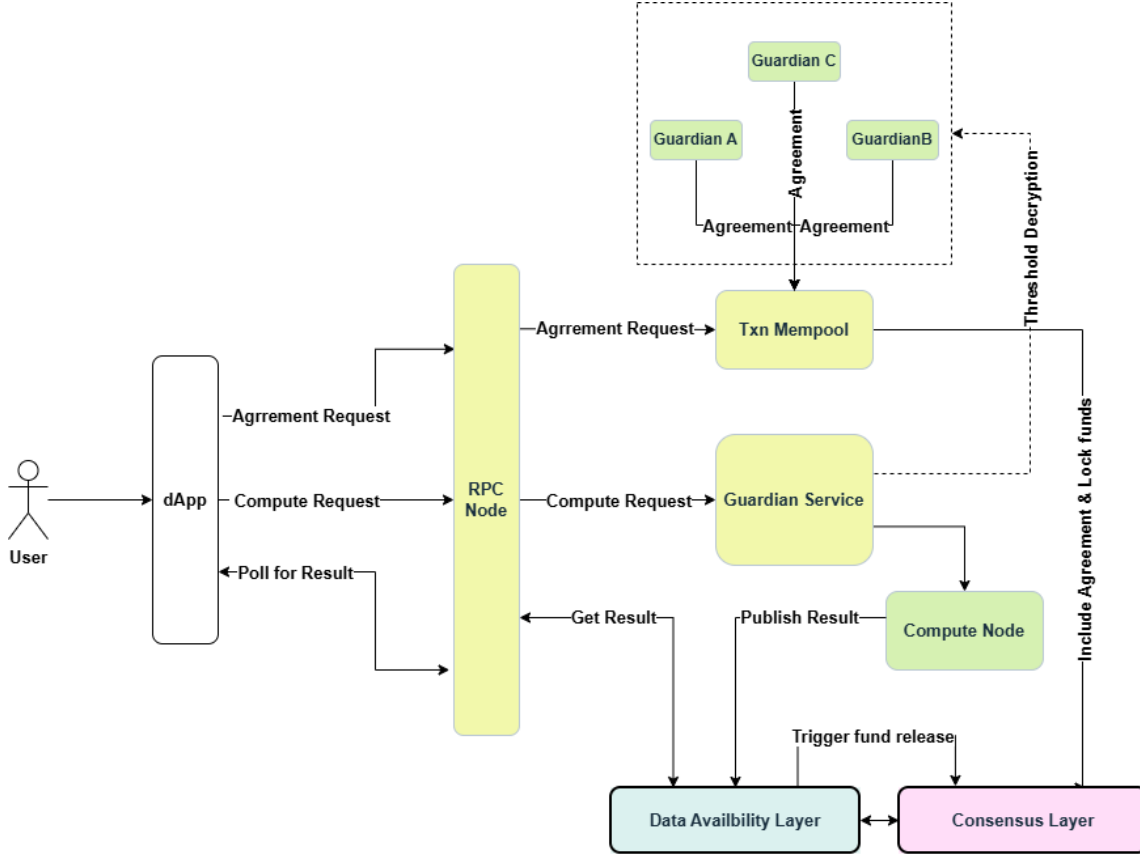


Figure 9: Compute Lifecycle

Further remarks

Manas supports also non-standard versions of Guardian Nodes such as Non-computing Trusted Nodes. A Non-computing Trusted Node is a Node that does not process the computation itself but delegates it to servers it trusts. Clearly, the communication with said servers is fully protected by cryptographic means (both from tampering and from leakage)

Manas also assigns a unique identifier to each Compute request to establish a connection among the contract, the request, and its result. In Manas, this identifier is generated by hashing key data points from the Compute request together with the CID. Thus, any dApp can independently compute and track this identifier in each block's event log, thereby ensuring deterministic linkage between the request and the result.

Performance. The Manas network comes with a runtime contract that enables 4 MB-sized blocks with a block time of 6.0 seconds. The block sizes can be increased in future to accommodate higher throughput. The current test-net network can reach a TPS of 700 tx/sec. The Manas testnet is running a BABE-GRANDPA protocol for block authoring, finality, and fork selection, providing a finality time in the range of 15-20 seconds. The DA Manas network incorporates an archive store of block history, as well as large-file storage. Currently, Manas storage node is configured for 300GB of raw storage, served by an erasure encoding-enabled database. Guardian clients can join the network every 4-hour interval.

The system’s performance is linked to the time for computations. A small overhead is present due to the transit of data between the node-client and the compute service. The overhead adds up to 1 or 2 block generation time. In the implementation, we have currently enabled support for a few AI models (Llama-3.1-8B-Instruct, DeepSeek-R1, DeepSeek-V3-0324, Qwen3-8B, and Llama-150m-fresh). The model takes about 15 seconds to generate a response of about 400 words. Obviously, the model’s performance greatly depends on the GPU’s performance. The EC2 instance has a Tesla T4 GPU, with 16 GB of memory, and a system configuration that includes 8 virtual CPU cores and 16 GB of RAM. This setup achieves token generation at approximately 40 tokens per second. For example, if a prompt is sent at block #1 and the generation time takes about 15 seconds, with a block time of 6 seconds, the results would appear on the Manas network in block #4.

5.3 Conclusions

The only way we can expect people to somehow share confidential and valuable data is within a digital ecosystem where every honest actor gets a fair portion of the value growth according to its own contribution (sharing, computing, verifying, etc.), while malevolent actors find it economic inconvenient to behave maliciously.

In this white paper we have proposed Palliora, which is a system with the sought-after properties, provided with strong privacy and confidentiality guarantees that are partly due to the use of cryptography and partly due to its economic security model. We have sketched some applications and use cases, we have described the technological components that we plan to use, we have hinted to a feasible implementation. While we continuously improve our systems by adding features and increasing efficiency, Palliora’s real potential will only be reached by its adoption and dapps creation. Indeed, Palliora is like a race track, ready to be used by powerful sleek cars, which are the dApps running on it. Therefore, we will need a standardized set of dApps, that can leverage fully Palliora capabilities. Among these, certainly we will need a tangle of dApps allowing the functioning of a *marketplace*. We will also need dApps specialized for the most promising use cases, such as those described in our examples throughout the white paper.

We believe the final result will be satisfactory and timely for current challenges in our transformative digital economy, including for the current “world race” to the development of AI platform, since Palliora allows safe and controlled training/inference of AI models that depends on the selected disclosure of confidential personal data.

References

- [1] OG Labs. *OG: Towards Data Availability 2.0*. 2024. URL: <https://og.ai> (visited on 2026-02-15).
- [2] M. Abdalla, F. Bourse, A. De Caro, and D. Pointcheval. “Simple Functional Encryption Schemes for Inner Products”. In: *PKC 2015: 18th International Conference on Theory and Practice of Public Key Cryptography*. Ed. by J. Katz. Vol. 9020. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, 2015-03, pp. 733–751. DOI: 10.1007/978-3-662-46447-2_33.
- [3] Apple Intelligence Preview. URL: <https://www.apple.com/apple-intelligence/> (visited on 2026-02-15).
- [4] Arcium. *Arcium*. URL: <https://www.arcium.com/> (visited on 2026-02-15).
- [5] G. Ateniese, R. C. Burns, R. Curtmola, J. Herring, L. Kissner, Z. N. J. Peterson, and D. Song. “Provable data possession at untrusted stores”. In: *ACM CCS 2007: 14th Conference on Computer and Communications Security*. Ed. by P. Ning, S. De Capitani di Vimercati, and P. F. Syverson. ACM Press, 2007-10, pp. 598–609. DOI: 10.1145/1315245.1315318.
- [6] G. Ateniese, L. Chen, M. Etemad, and Q. Tang. “Proof of Storage-Time: Efficiently Checking Continuous Data Availability”. In: *ISOC Network and Distributed System Security Symposium – NDSS 2020*. The Internet Society, 2020-02. DOI: 10.14722/ndss.2020.24427.
- [7] G. Ateniese, S. Kamara, and J. Katz. “Proofs of Storage from Homomorphic Identification Protocols”. In: *Advances in Cryptology – ASIACRYPT 2009*. Ed. by M. Matsui. Vol. 5912. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, 2009-12, pp. 319–333. DOI: 10.1007/978-3-642-10366-7_19.
- [8] Avail Team. *Avail: A Unifying Blockchain Network*. Version 2.1. 2024. URL: <https://www.availproject.org> (visited on 2026-02-15).
- [9] S. Balaji, M. N. Krishnan, M. Vajha, V. Ramkumar, B. Sasidharan, and P. V. Kumar. “Erasure coding for distributed storage: An overview”. In: *Science China Information Sciences* 61 (2018), pp. 1–45.
- [10] M. Al-Bassam. *LazyLedger: A Distributed Data Availability Ledger With Client-Side Smart Contracts*. 2019. URL: <https://arxiv.org/abs/1905.09274>.
- [11] M. Al-Bassam, A. Sonnino, and V. Buterin. *Fraud and Data Availability Proofs: Maximising Light Client Security and Scaling Blockchains with Dishonest Majorities*. 2019. URL: <https://arxiv.org/abs/1809.09044>.
- [12] M. Al-Bassam, A. Sonnino, V. Buterin, and I. Khoffi. “Fraud and Data Availability Proofs: Detecting Invalid Blocks in Light Clients”. In: *FC 2021: 25th International Conference on Financial Cryptography and Data Security, Part II*. Ed. by N. Borisov and C. Díaz. Vol. 12675. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, 2021-03, pp. 279–298. DOI: 10.1007/978-3-662-64331-0_15.
- [13] J. C. Benaloh. “Secret sharing homomorphisms: Keeping shares of a secret secret”. In: *Conference on the Theory and Application of Cryptographic Techniques*. Berlin: Springer-Verlag, 1987.
- [14] Bitcoin P2P e-cash paper. URL: <https://www.metzdowd.com/pipermail/cryptography/2008-October/014810.html> (visited on 2026-02-15).
- [15] D. Boneh, A. Sahai, and B. Waters. “Functional Encryption: Definitions and Challenges”. In: *TCC 2011: 8th Theory of Cryptography Conference*. Ed. by Y. Ishai. Vol. 6597. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, 2011-03, pp. 253–273. DOI: 10.1007/978-3-642-19571-6_16.
- [16] S. Bowe, A. Chiesa, M. Green, I. Miers, P. Mishra, and H. Wu. “ZEXE: Enabling Decentralized Private Computation”. In: *2020 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, 2020-05, pp. 947–964. DOI: 10.1109/SP40000.2020.00050. URL: <https://eprint.iacr.org/2018/962>.
- [17] C. Brunetta, A. Chaudhary, S. Galatolo, and M. Sala. *An Approach to Computable Contracts with Verifiable Computation Outsourcing and Blockchain Transactions*. Cryptology ePrint Archive, Report 2025/1896. 2025. URL: <https://eprint.iacr.org/2025/1896>.
- [18] C. Brunetta, A. Chaudhary, S. Galatolo, and M. Sala. *Stabilizing the Staking Rate, Dynamically Distributed Inflation and Delay Induced Oscillations*. 2025. DOI: 10.48550/ARXIV.2510.11065.
- [19] C. Brunetta, A. Chaudhary, S. Galatolo, and M. Sala. “Stabilizing the Staking Rate, Dynamically Distributed Inflation and Delay Induced Oscillations”. In: *Financial Crypto 2026 (to appear)*. Lecture Notes in Computer Science. Springer, 2026.
- [20] C. Brunetta, A. Chaudhary, S. Galatolo, and M. Sala. “Work-in-Progress: an Approach to Computable Contracts with Verifiable Computation Outsourcing and Blockchain Transactions”. In: *Financial Cryptography and Data Security. FC 2026 International Workshops (to appear)* (2026).

- [21] C. Brunetta and M. Sala. “SoK: Modelling Data Storage and Availability”. In: *Financial Cryptography and Data Security. FC 2025 International Workshops* (2025). DOI: 10.1007/978-3-032-00492-5_17.
- [22] E. Budish, A. Lewis-Pye, and T. Roughgarden. “The Economic Limits of Permissionless Consensus”. In: *Proceedings of the 25th ACM Conference on Economics and Computation. EC ’24*. Association for Computing Machinery, 2024-07, pp. 704–731. DOI: 10.1145/3670865.3673548. URL: <https://arxiv.org/abs/2405.09173>.
- [23] B. Bünz, S. Agrawal, M. Zamani, and D. Boneh. “Zether: Towards Privacy in a Smart Contract World”. In: *FC 2020: 24th International Conference on Financial Cryptography and Data Security*. Ed. by J. Bonneau and N. Heninger. Vol. 12059. Lecture Notes in Computer Science. Springer, Cham, 2020-02, pp. 423–443. DOI: 10.1007/978-3-030-51280-4_23. URL: <https://eprint.iacr.org/2019/191>.
- [24] Celestia Labs. *Celestia*. 2025. URL: <https://celestia.org> (visited on 2026-02-15).
- [25] A. Cevallos and J. Gehrlein. *Token Economics*. URL: <https://research.web3.foundation/Polkadot/overview/token-economics> (visited on 2026-02-15).
- [26] A. Chaudhary. “Decentralized Finance: Labyrinth”. In: *De Cifris Koine 5* (2025-02), pp. 76–77. DOI: 10.69091/koine/vol-5-W05.
- [27] Z. Cheng et al. *OML: Open, Monetizable, and Loyal AI*. 2024. URL: <https://arxiv.org/abs/2411.03887>.
- [28] L. Cong, Z.-F. He, and K. Tang. “Staking, Token Pricing, and Crypto Carry”. In: *Cornell University, Johnson Graduate School of Management Research Paper* (2022).
- [29] L. W. Cong, Z. He, and K. Tang. “The Tokenomics of Staking”. In: (2025-04). DOI: 10.3386/w33640.
- [30] S. Deb, R. Raynor, and S. Kannan. *STAKESURE: Proof of Stake Mechanisms with Strong Cryptoeconomic Safety*. 2024. URL: <https://arxiv.org/abs/2401.05797>.
- [31] *DFINITY*. URL: <https://dfinity.org/> (visited on 2026-02-15).
- [32] P. Dixit, A. K. Gupta, M. C. Trivedi, and V. K. Yadav. “Traditional and hybrid encryption techniques: a survey”. In: *Networking Communication and Data Knowledge Engineering: Volume 2*. Springer, 2018, pp. 239–248.
- [33] T. V. T. Doan, M.-L. Messai, G. Gavin, and J. Darmont. “A survey on implementations of homomorphic encryption schemes”. In: *The Journal of Supercomputing* 79.13 (2023), pp. 15098–15139.
- [34] EigenCloud. *EigenCloud*. URL: <https://www.eigencloud.xyz> (visited on 2026-02-15).
- [35] *EigenDA*. URL: <https://www.eigenda.xyz/> (visited on 2026-02-15).
- [36] *Ethereum Whitepaper*. URL: <https://ethereum.org/en/whitepaper/> (visited on 2026-02-15).
- [37] European Parliament. Directorate General for Internal Policies of the Union. *Challenges and limits of an open source approach to Artificial Intelligence*. Publications Office, 2021. DOI: 10.2861/971700. (Visited on 2026-02-15).
- [38] M. Ezekiel. “The Cobweb Theorem”. In: *The Quarterly Journal of Economics* 52.2 (1938), pp. 255–280.
- [39] A. Gabizon, Z. J. Williamson, and O. Ciobotaru. *PLONK: Permutations over Lagrange-bases for Oecumenical Noninteractive arguments of Knowledge*. Cryptology ePrint Archive, Report 2019/953. 2019. URL: <https://eprint.iacr.org/2019/953>.
- [40] S. Garg, D. Kolonelos, G.-V. Policharla, and M. Wang. “Threshold Encryption with Silent Setup”. In: *Advances in Cryptology – CRYPTO 2024, Part VII*. Ed. by L. Reyzin and D. Stebila. Vol. 14926. Lecture Notes in Computer Science. Springer, Cham, 2024-08, pp. 352–386. DOI: 10.1007/978-3-031-68394-7_12. URL: <https://eprint.iacr.org/2024/263>.
- [41] R. Geelen and F. Vercauteren. “Fully Homomorphic Encryption for Cyclotomic Prime Moduli”. In: *Advances in Cryptology – EUROCRYPT 2025, Part III*. Ed. by S. Fehr and P.-A. Fouque. Vol. 15603. Lecture Notes in Computer Science. Springer, Cham, 2025-05, pp. 366–397. DOI: 10.1007/978-3-031-91131-6_13.
- [42] Gensyn. *Gensyn*. URL: <https://www.gensyn.ai/> (visited on 2026-02-15).
- [43] E. Gent. *The tech industry can’t agree on what open-source AI means. That’s a problem*. 2024-03-25. URL: <https://www.technologyreview.com/2024/03/25/1090111/tech-industry-open-source-ai-definition-problem> (visited on 2026-02-15).
- [44] E. Gibney. “Not all ‘open source’ AI models are actually open: here’s a ranking”. In: *Nature* (2024-06). DOI: 10.1038/d41586-024-02012-5. URL: <http://dx.doi.org/10.1038/d41586-024-02012-5>.

- [45] *Grammarly: AI Writing Assistance*. URL: <https://www.grammarly.com/> (visited on 2026-02-15).
- [46] S. Gurtler and I. Goldberg. “SoK: Privacy-Preserving Reputation Systems”. In: *Proceedings on Privacy Enhancing Technologies* 2021.1 (2021-01), pp. 107–127. DOI: 10.2478/popets-2021-0007.
- [47] U. Haböck, D. Levit, and S. Papini. *Circle STARKs*. Cryptology ePrint Archive, Report 2024/278. 2024. URL: <https://eprint.iacr.org/2024/278>.
- [48] S. Halevi, D. Harnik, B. Pinkas, and A. Shulman-Peleg. “Proofs of ownership in remote storage systems”. In: *ACM CCS 2011: 18th Conference on Computer and Communications Security*. Ed. by Y. Chen, G. Danezis, and V. Shmatikov. ACM Press, 2011-10, pp. 491–500. DOI: 10.1145/2046707.2046765.
- [49] F. Hendrikx, K. Bubendorfer, and R. Chard. “Reputation systems: A survey and taxonomy”. In: *Journal of Parallel and Distributed Computing* 75 (2015-01), pp. 184–197. DOI: 10.1016/j.jpdc.2014.08.004.
- [50] P. Jauernig, A.-R. Sadeghi, and E. Stapf. “Trusted execution environments: properties, applications, and challenges”. In: *IEEE Security & Privacy* 18.2 (2020), pp. 56–60.
- [51] N. Kaldor. “A Classificatory Note on the Determinateness of Equilibrium”. In: *The Review of Economic Studies* 1.2 (1934), pp. 122–136.
- [52] H. Kalodner, S. Goldfeder, X. Chen, S. M. Weinberg, and E. W. Felten. “Arbitrum: Scalable, private smart contracts”. In: *27th USENIX Security Symposium (USENIX Security 18)*. Baltimore, MD: USENIX Association, 2018-08, pp. 1353–1370. URL: <https://www.usenix.org/conference/usenixsecurity18/presentation/kalodner>.
- [53] A. Kate, G. M. Zaverucha, and I. Goldberg. “Constant-Size Commitments to Polynomials and Their Applications”. In: *Advances in Cryptology – ASIACRYPT 2010*. Ed. by M. Abe. Vol. 6477. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, 2010-12, pp. 177–194. DOI: 10.1007/978-3-642-17373-8_11.
- [54] J. Leshno, R. Pass, and E. Shi. *Can open decentralized ledgers be economically secure?* Cryptology ePrint Archive, Report 2023/1516. 2023. URL: <https://eprint.iacr.org/2023/1516>.
- [55] C. Li, M. Xu, J. Zhang, H. Guo, and X. Cheng. “SoK: Decentralized storage network”. In: *High-Confidence Computing* 4.3 (2024), p. 100239. DOI: <https://doi.org/10.1016/j.hcc.2024.100239>.
- [56] J. Liang, D. Hu, P. Wu, Y. Yang, Q. Shen, and Z. Wu. “SoK: understanding zk-SNARKs: the gap between research and practice”. In: *Proceedings of the 34th USENIX Conference on Security Symposium*. USA: USENIX Association, 2025. URL: <https://eprint.iacr.org/2025/172>.
- [57] J. Lin and Z. Gao. “The Vision of Secure Multi-party Computation Technical Standards”. In: *International Conference on Research in Security Standardisation*. Springer. 2024, pp. 17–30.
- [58] C. Marcolla, V. Sucasas, M. Manzano, R. Bassoli, F. H. P. Fitzek, and N. Aaraj. “Survey on Fully Homomorphic Encryption, Theory, and Applications”. In: *Proceedings of the IEEE* 110.10 (2022-10), pp. 1572–1609. DOI: 10.1109/jproc.2022.3205665. URL: <https://eprint.iacr.org/2022/1602>.
- [59] C. Mascia, M. Sala, and I. Villa. “A survey on functional encryption”. In: *Advances in Mathematics of Communications* 17.5 (2023), pp. 1251–1289. DOI: 10.3934/amc.2021049.
- [60] T. Moran and I. Orlov. *Proofs of Space-Time and Rational Proofs of Storage*. Cryptology ePrint Archive, Report 2016/035. 2016. URL: <https://eprint.iacr.org/2016/035>.
- [61] Movement Labs. *Movement Network: High-Throughput Fast Finality Move-based Chains Secured by Ethereum*. Version 0.2.7. 2025. URL: https://www.movementnetwork.xyz/whitepaper/movement-whitepaper_en.pdf (visited on 2026-02-15).
- [62] A. Muñoz, R. Ríos, R. Román, and J. López. “A survey on the (in)security of trusted execution environments”. In: *Computers & Security* 129 (2023-06), p. 103180. DOI: 10.1016/j.cose.2023.103180.
- [63] S. Nakamoto. “Bitcoin: A Peer-to-Peer Electronic Cash System”. In: *Cryptography Mailing list* (2008-10). URL: <https://metzdowd.com>.
- [64] National Institute of Standards and Technology. *Multi-Party Threshold Cryptography (MPTC)*. URL: <https://csrc.nist.gov/projects/threshold-cryptography> (visited on 2026-02-15).
- [65] National Institute of Standards and Technology. *Post-Quantum Cryptography (PQC)*. URL: <https://csrc.nist.gov/projects/post-quantum-cryptography> (visited on 2026-02-15).
- [66] Near AI. *Near AI*. URL: <https://www.near.org/> (visited on 2026-02-15).
- [67] Nillion. *Nillion*. URL: <https://nillionhub.org/> (visited on 2026-02-15).

- [68] Nouns Research. *Nouns Research*. URL: <https://nounsresearch.com/> (visited on 2026-02-15).
- [69] OpenAI. *Video generation models as world simulators*. URL: <https://openai.com/index/video-generation-models-as-world-simulators> (visited on 2026-02-15).
- [70] OpenGravity. *OpenGravity*. URL: <https://opengravity.com/> (visited on 2026-02-15).
- [71] E. Orsini. “Efficient, Actively Secure MPC with a Dishonest Majority: A Survey”. In: *Arithmetic of Finite Fields*. Springer International Publishing, 2021, pp. 42–71. DOI: 10.1007/978-3-030-68869-1_3. URL: <https://eprint.iacr.org/2022/417>.
- [72] Partisia Blockchain. *Partisia Blockchain*. URL: <https://partisiablockchain.com> (visited on 2026-02-15).
- [73] Polkadot. *Polkadot: Instruments for Individuality*. URL: <https://polkadot.com>.
- [74] Protocol Labs. *Filecoin: A Decentralized Storage Network*. 2017. URL: <https://filecoin.io/filecoin.pdf> (visited on 2026-02-15).
- [75] E. A. Quaglia. *DeCifrisTrends Lecture 9 "Protocols for Peer Rating Systems"*. 2023. URL: <https://www.youtube.com/watch?v=HWp-KfHRvKQ>.
- [76] Ritual. *Ritual*. URL: <https://ritual.net/> (visited on 2026-02-15).
- [77] M. Sabt, M. Achemlal, and A. Bouabdallah. “Trusted execution environment: What it is, and what it is not”. In: *2015 IEEE Trustcom/BigDataSE/Ispa*. Vol. 1. IEEE. 2015, pp. 57–64.
- [78] Sahara Labs. *Sahara Labs : Litepaper*. URL: <https://saharalabs.ai/learn/litepaper#introduction> (visited on 2026-02-15).
- [79] B. Sengupta, S. Bag, S. Ruj, and K. Sakurai. “Retricoin: Bitcoin based on compact proofs of retrievability”. In: *Proceedings of the 17th International Conference on Distributed Computing and Networking*. New York, NY, USA: Association for Computing Machinery, 2016. DOI: 10.1145/2833312.2833317.
- [80] Sentient. *Sentient*. URL: <https://sentient.foundation/> (visited on 2026-02-15).
- [81] A. Shamir. “How to Share a Secret”. In: *Commun. ACM* 22.11 (1979), pp. 612–613. DOI: 10.1145/359168.359176.
- [82] A. Singh, C. Lu, G. Gupta, A. Chopra, J. Blanc, T. Klinghoffer, K. Tiwary, and R. Raskar. *A Perspective on Decentralizing AI*. 2024-07. URL: <https://www.media.mit.edu/publications/decai-perspective/>.
- [83] A. Skidanov, I. Polosukhin, and B. Wang. *Nightshade: Near Protocol Sharding Design 2.0*. 2024. URL: <https://near.org/papers> (visited on 2026-02-15).
- [84] *Stability AI*. URL: <https://stability.ai/> (visited on 2026-02-15).
- [85] StarkWare. *ethSTARK Documentation*. Cryptology ePrint Archive, Report 2021/582. 2021. URL: <https://eprint.iacr.org/2021/582>.
- [86] *Substrate Blockchain Technology*. URL: <https://substrate.io/technology/> (visited on 2026-02-15).
- [87] L. T. Thibault, T. Sarry, and A. S. Hafid. “Blockchain Scaling Using Rollups: A Comprehensive Survey”. In: *IEEE Access* 10 (2022), pp. 93039–93054. DOI: 10.1109/ACCESS.2022.3200051.
- [88] H. Touvron et al. *LLaMA: Open and Efficient Foundation Language Models*. 2023. URL: <https://arxiv.org/abs/2302.13971>.
- [89] Vana Protocol. *Vana Protocol*. URL: <https://www.vana.org/> (visited on 2026-02-15).
- [90] D. Vorick and L. Champine. *Sia: Simple Decentralized Storage*. 2014. URL: <https://sia.tech/sia.pdf> (visited on 2026-02-15).
- [91] S. Wilkinson, T. Boshevski, J. Brandoff, J. Prestwich, G. Hall, P. Gerbes, P. Hutchins, C. Pollard, and V. Buterin. *Storj: A Peer-to-Peer Cloud Storage Network*. 2016. URL: <https://storj.io/storjv2.pdf> (visited on 2026-02-15).
- [92] S. Williams, A. Kedia, L. Berman, and S. Campos-Groth. *Arweave: The Permanent Information Storage Protocol*. 2023. URL: <https://arweave.org> (visited on 2026-02-15).
- [93] I. Wladawsky-Berger. *Open Source AI: Opportunities and Challenges*. 2024-02-09. URL: <https://www.linuxfoundation.org/blog/open-source-ai-opportunities-and-challenges> (visited on 2026-02-15).
- [94] J. Xu, A. Yang, J. Zhou, and D. S. Wong. *Lightweight and Privacy-Preserving Delegatable Proofs of Storage*. Cryptology ePrint Archive, Report 2014/395. 2014. URL: <https://eprint.iacr.org/2014/395>.

-
- [95] A. Yang, J. Xu, J. Weng, J. Zhou, and D. S. Wong. “Lightweight and Privacy-Preserving Delegatable Proofs of Storage with Data Dynamics in Cloud Storage”. In: *IEEE Transactions on Cloud Computing* 9.1 (2021-01), pp. 212–225. DOI: 10.1109/tcc.2018.2851256.
 - [96] R. Yang, Z. Yu, and W. Susilo. “Fully Homomorphic Encryption with Chosen-Ciphertext Security from LWE”. In: *Advances in Cryptology – CRYPTO 2025, Part III*. Ed. by Y. T. Kalai and S. F. Kamara. Vol. 16002. Lecture Notes in Computer Science. Springer, Cham, 2025-08, pp. 371–405. DOI: 10.1007/978-3-032-01881-6_12.
 - [97] N. Zahed Benisi, M. Aminian, and B. Javadi. “Blockchain-based decentralized storage networks: A survey”. In: *Journal of Network and Computer Applications* 162 (2020), p. 102656. DOI: <https://doi.org/10.1016/j.jnca.2020.102656>.
 - [98] C. Zhang, X. Li, and M. H. Au. “ePoSt: Practical and Client-Friendly Proof of Storage-Time”. In: *IEEE Transactions on Information Forensics and Security* 18 (2023), pp. 1052–1063. DOI: 10.1109/TIFS.2022.3233780.
 - [99] I. Zhou, F. Tofigh, M. Piccardi, M. Abolhasan, D. Franklin, and J. Lipman. “Secure multi-party computation for machine learning: A survey”. In: *IEEE Access* (2024). DOI: 10.1109/ACCESS.2024.3388992.

A Stabilization Mechanism for the Staking Rate

The following appendix is an ad-hoc explanation extracted from the published article by part of the authors [19] (or via pre-print [18]). Differently from the published papers, this appendix considers specific desired inflation range/parameters (i.e. different values more coherent with Palliora’s application), showcase the stabilization mechanism of Palliora and limits its comparison against Polkadot. We point the interested reader to the published paper for full explanation of the phenomenon, formal analysis and further comparison with a “fixed inflation” curve.

Many blockchains based on Proof-of-Stake (PoS) protocols need to maintain the *staking rate* within a target range, depending on the blockchain itself. For example, a staking rate that is too low can compromise network security, while an excessively high rate can harm the token’s liquidity and the health of the transactional ecosystem. To achieve this balance, protocols such as Polkadot have implemented reward mechanisms that tie stakers’ rewards to token inflation, following a non-linear curve designed to maximize the inflation distributed to stakers at the target *staking rate* [25, 28] (see Figure 10).

While this approach is effective in creating a strong incentive towards an equilibrium point, its architecture presents dynamical vulnerabilities. Specifically, the yield curve typically exhibits a very steep gradient (high sensitivity) around the target point. A small deviation in the *staking rate* causes a sharp change in the yield, generating an excessive corrective signal for economic agents.

This high sensitivity, when coupled with the *delayed feedback* with which stakers react to yield changes, creates the conditions for oscillations. Economic literature has identified how the combination of high reactivity and decisional lags can generate cycles and instability in dynamical systems [38, 51].

In the context of staking, this delay is both structural (due, for example, to unbonding periods) and behavioural. Recent empirical evidence confirms this dynamic, showing that the reward rate in one period is a significant predictor of an increase in the *staking rate* in the next period [29]. This delayed reaction by agents to a highly-volatile market signal can cause the system to continuously over-correct, oscillating around the desired equilibrium rather than converging to it stably.

To mitigate this potential instability, this section proposes an alternative inflation management mechanism. The objective is to replace the concept of an “inflation peak” with a more robust “stability corridor”, thereby nullifying the yield’s sensitivity within a target range while maintaining strong corrective incentives outside of it.

Our model is based on a piecewise-affine total inflation function, which defines the quantity of new tokens issued as a function of the total *staking rate*, denoted by $\sigma \in [0, 1]$. The total inflation, $I(\sigma)$, which also corresponds to the total rewards distributed to stakers, is defined by three distinct linear regimes designed to govern the behaviour of economic agents.

We now describe the inflation function we propose and motivate our choices. We assume a target range for the *staking rate* between 40% and 60%. We want the inflation to range between a minimum of 3% to a maximum of 6% and to vary continuously with three affine branches. The inflation function $I(\sigma)$ we propose is defined as follows:

$$I(\sigma) = \begin{cases} 0.03 + 0.025 \cdot \sigma & \text{if } 0 \leq \sigma < 0.4 \\ 0.1 \cdot \sigma & \text{if } 0.4 \leq \sigma \leq 0.6 \\ 0.105 - 0.075 \cdot \sigma & \text{if } 0.6 < \sigma \leq 1.0 \end{cases} \quad (1)$$

Figure 10 graphically illustrates the proposed distributed-inflation function and the relative Annual Percentage Yield (APY). As the figure clearly shows, the behaviour of such inflation system is characterized by three different phases:

1. **Incentivization Phase** ($0 \leq \sigma < 0.4$): for low levels of participation, it is crucial to incentivize new stakers. Inflation starts from a base value I_{base} (set to 3%) and grows linearly to reach $I(0.4) = 4\%$. The associated yield for token invested is very high, promoting the investment, until it stabilizes to 10% when $\sigma = 40\%$.

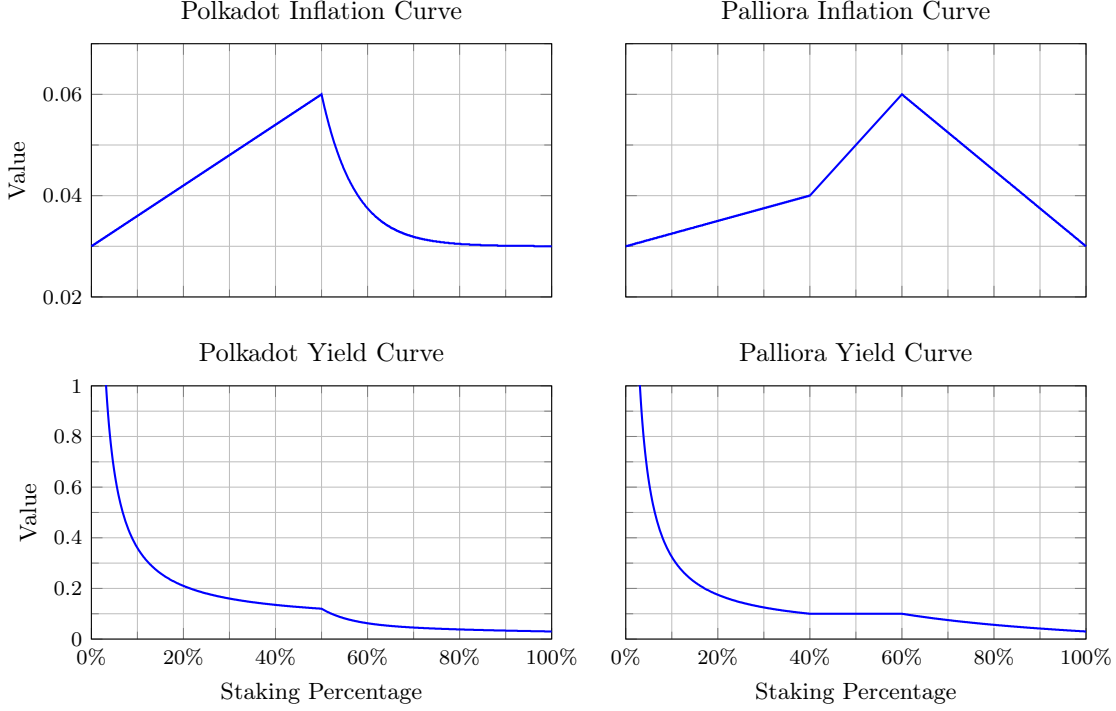


Figure 10: The total inflation curve $I(x)$ as a function of the staking rate x for both Polkadot and Equation (1). The function consists of three affine branches that define the incentivization, stability, and disincentivization phases. [19]

2. **Stability Corridor** ($0.4 \leq \sigma \leq 0.6$): in this interval, inflation continues to grow linearly, rising from 4% to its maximum of 6% at $\sigma = 0.6$. This linear growth is calibrated to make the individual yield constant to 10%. This is the core of our stabilization mechanism. By nullifying the change in yield, the sensitivity that, combined with delay, generates instability is eliminated. Agents have no incentive to alter their positions in response to fluctuations of the *staking rate* within this zone, thus promoting a stable equilibrium.
3. **Disincentivization Phase** ($0.6 < \sigma \leq 1.0$): to prevent excessive token illiquidity, after surpassing 60% staking, inflation decreases linearly from its 6% peak back to the base value $I_{\text{base}} = 3\%$ when $\sigma = 1.0$, and the yield also decreases similarly, discouraging further staking.

We illustrate the stabilization effect of the plateau by implementing a simple model of staking dynamics, where the users potentially interested to invest in staking react with a 7 days delay to the current staking rate and relative yield. In the model, the typical user would start its dynamics with a low staking (1%). The user would decide to increase its staking by 7% ,if the current yield is above 11.5%, to reduce its staking by the same amount if the yield is instead below 8.5% (but the action is going to be implemented with a 7 days delay). Otherwise, in the yields range 8.5 – 11.5%, the user would keep its current staking.

We compare the resulting dynamics for the staking rate obtained by our proposed dynamical-inflation function Equation (1) with the dynamics obtained implementing the Polkadot dynamical-inflation function with the same parameters (same minimum and maximum inflation, target staking rate set at 50%; the function and the associated yield can be seen in Figure 10). In Figure 11, we can see both the dynamics of the staking rate with the dynamical-inflation function *and* the dynamics obtained with the Polkadot-like dynamical-inflation function. From the picture, it is evident that, despite the two inflation functions having similar basic parameters, the presence of the stability corridor in Equation (1) does stabilize the stacking rate.

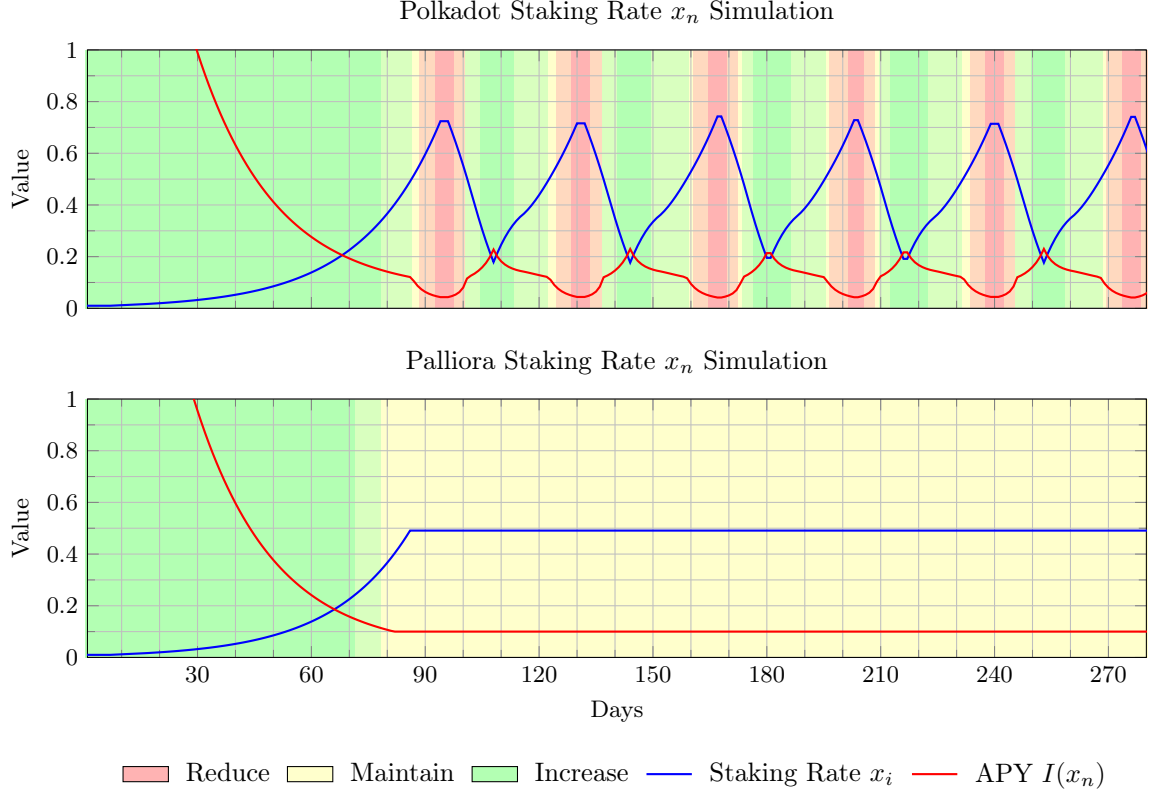


Figure 11: The dynamics of the staking rate with a Polkadot-like inflation. In blue we plot the staking rate as a function of time, in red the relative yield. The actions taken by the staker (increase, maintain or reduce staking) are plotted as background colour.

These simulations, are meant only to qualitatively illustrate the stabilization effect of the dynamical-inflation function Equation (1). While the 7 days delay reflects the average delay observed by Cong et al. [29], the other parameters have been chosen to reflect a reasonable agents behaviour, but they are not supported by systematic empirical studies. We have added our source code in Section A.1. In Pallora these parameters will be automatically measured by the statistics coming from the users behaviour, so that they can be used to adapt the dynamical-inflation function to the current economical state of the system.

Conclusions. We believe that the approach above described offers several significant advantages. First, it eliminates excessive sensitivity by creating a yield plateau, nullifying the incentive for stakers to react to minor fluctuations within the corridor (which in turn dampens the oscillations caused by delayed feedback). Establishing a “stability corridor” (e.g., 40%–60%), rather than a fragile single equilibrium point, allows a higher resilience of the system to shocks. Furthermore, the resulting predictability of returns *reduces speculative behaviour* by lowering uncertainty for stakers, thus attracting participants with a long-term orientation. Finally, this model somewhat simplifies governance, as network parameters are reduced to defining the boundaries of the corridor and the target yield, rather than calibrating a complex non-linear curve.

A.1 Simulation Source Code

All the source code provided is written in Julia.

Polkadot: Inflation and APY Curve

```

1 # Parameters
2 chi_ideal = 0.5
3 i_ideal = 0.12
4 I_0 = 0.03
5 d = 0.05
6
7 # Inflation Curve
8 function I_NPoS(x; chi_ideal=chi_ideal, i_ideal=i_ideal, I_0=I_0, d=d)
9     I_ideal = chi_ideal * i_ideal
10    if x <= chi_ideal
11        return I_0 + ((I_ideal - I_0) / chi_ideal) * x
12    else
13        return I_0 + (I_ideal - I_0) * 2^((chi_ideal - x) / d)
14    end
15 end
16
17 # Defining APY as I_NPoS(x)/x
18 function APY(x)
19     return I_NPoS(x) / x
20 end
21
22 # Generate x values for curves
23 # Avoid x = 0 in APY
24 xs_APY = 0.001:0.001:1.0
25 xs = 0.0:0.001:1.0
26
27 # Inflation Curve and Yield Curve
28 ys = [I_NPoS(x) for x in xs]
29 apy_values = [APY(x) for x in xs_APY]

```

Palliora: Inflation (Equation (1)) and APY Curve

```

1 # Parameters
2 chi_ideal = 0.5
3 i_ideal = 0.12
4 I_0 = 0.03
5 d = 0.05
6
7 # Inflation Curve
8 function I_NPoS(x; chi_ideal=chi_ideal, i_ideal=i_ideal, I_0=I_0, d=d)
9     I_max = chi_ideal * i_ideal
10    x_low = 0.4
11    x_up = 0.6
12    # To guarantee continuity in x_low
13    I_1 = (I_max / x_up) * x_low
14    if x <= x_low
15        # Linear Increment between I_0 and I_1
16        return I_0 + ((I_1 - I_0) / x_low)*x
17    elseif x <= x_up
18        # Stability Corridor
19        return (I_max / x_up) * x
20    else
21        # Linear Decrease
22        return I_max + ((I_0 - I_max) / (1 - x_up))*(x - x_up)
23    end
24 end
25
26 # Defining APY as I_NPoS(x)/x
27 function APY(x)
28     return I_NPoS(x) / x
29 end
30
31 # Generate x values for curves
32 # Avoid x = 0 in APY
33 xs_APY = 0.001:0.001:1.0
34 xs = 0.0:0.001:1.0

```

```

35
36 # Inflation Curve and Yield Curve
37 ys = [I_NPoS(x) for x in xs]
38 apy_values = [APY(x) for x in xs_apy]

```

Polkadot: Simulation

```

1 # Base parameters: ideal staking rate, top inflation, base inflation, exponential decay rate (
  only in Polkadot)
2 chi_ideal = 0.5
3 i_ideal = 0.12
4 I_0 = 0.03
5 d = 0.05
6
7 # Inflation function, distributed to stakers
8 function I_NPoS(x; chi_ideal=chi_ideal, i_ideal=i_ideal, I_0=I_0, d=d)
9     I_ideal = chi_ideal * i_ideal
10    if x <= chi_ideal
11        return I_0 + ((I_ideal - I_0) / chi_ideal) * x
12    else
13        return I_0 + (I_ideal - I_0) * 2^((chi_ideal - x) / d)
14    end
15 end
16
17 # APY (yield)
18 function APY(x)
19     return I_NPoS(x) / x
20 end
21
22 # Dynamical simulation, and parameters threshold=when they buy or sell, x0=initial staking
23 function simulate_staking(; days=280, delay=7, change=0.07, threshold=0.1, x0=0.01)
24     x_vals = fill(x0, days)
25     actions = zeros(days)
26
27     for n in delay:(days- delay)
28         x_current = x_vals[n]
29         apy = APY(x_current)
30
31         # Action is chosen based on the threshold
32         if apy > threshold + 0.015
33             actions[n + delay] += +1 #apy - threshold
34         elseif apy < threshold - 0.015
35             actions[n + delay] += -1 #apy - threshold
36         end
37
38
39
40         # Decided action is applied to the staking
41         if n + delay <= days
42             dx = actions[n] * change * x_vals[n-delay+1]
43             x_vals[n+1] = clamp(x_vals[n] + dx, 0.01, 0.99)
44         end
45     end
46
47     return x_vals, actions
48 end
49
50 # Run simulation
51 days = 300
52 x_sim, actions = simulate_staking(days=days)
53
54 # Compute APY every day
55 apy_sim = [APY(x) for x in x_sim]

```

Palliora: Simulation

```

1 # Base parameters: ideal staking rate, top inflation, base inflation, exponential decay rate (
  only in Polkadot)
2 chi_ideal = 0.5
3 i_ideal = 0.12

```

```

4  I_0 = 0.03
5  d = 0.05
6
7  # Inflation function, distributed to stakers
8  function I_NPoS(x; i_ideal=i_ideal, I_0=I_0, d=d)
9      I_max = chi_ideal * i_ideal
10     # Stability interval
11     x_low = 0.4
12     x_up = 0.6
13     # Computations to make the inflation to be continuous
14     I_1 = (I_max / x_up) * x_low
15     if x <= x_low
16         # First branch x in [0,0.4]
17         return I_0 + ((I_1 - I_0)/x_low) * x
18     elseif x <= x_up
19         # Second branch: function of the type C*x, with C = I_max/0.6
20         return (I_max / x_up) * x
21     else
22         # Third branch: linear decrease from I_max to I_0 for x in (0.6,1]
23         return I_max + ((I_0 - I_max)/(1 - x_up)) * (x - x_up)
24     end
25 end
26
27 # APY (yield)
28 function APY(x)
29     return I_NPoS(x) / x
30 end
31
32 # Dynamical simulation, and parameters threshold=when they buy or sell, x0 = initial staking
33 function simulate_staking(; days=180, delay=7, change=0.07, threshold=0.1, x0=0.01)
34     x_vals = fill(x0, days)
35     actions = zeros(days)
36
37     for n in delay:(days - delay)
38         x_current = x_vals[n]
39         apy = APY(x_current)
40
41         # Action is chosen based on the threshold
42         if apy > threshold + 0.015
43             actions[n + delay] += +1 #apy - threshold
44         elseif apy < threshold - 0.015
45             actions[n + delay] += -1 #apy - threshold
46         end
47
48
49
50         # Decided action is applied to the staking
51         if n + delay <= days
52             dx = actions[n] * change * x_vals[n-delay+1]
53             x_vals[n+1] = clamp(x_vals[n] + dx, 0.01, 0.99)
54         end
55     end
56
57     return x_vals, actions
58 end
59
60 # Run simulation
61 days = 300
62 x_sim, actions = simulate_staking(days=days)
63
64 # Compute APY every day
65 apy_sim = [APY(x) for x in x_sim]

```

Change Log

- Version 1.1**
- Improved presentation thanks to the reviews by Nigel Smart and Massimo Morini.
 - More detailed presentation on Palliora contracts and protocol.
 - Refined technical description thanks to published papers [19, 20].
 - References improvements and updated hyperlinks' access availability.
 - Minor grammatical corrections and general reading improvements.
 - Added document's acknowledgment.
- Version 1.0**
- Initial version of this document.

List of Acronyms

- ▶ AI: Artificial Intelligence.
- ▶ DA: Data Availability.
- ▶ DAA: Data Availability and Accountability.
- ▶ DAC: Data Availability and Computability.
- ▶ dApps: Decentralized Applications.
- ▶ EVM: Ethereum Virtual Machine.
- ▶ FPC: Fee Pricing Committee.
- ▶ FE: Functional Encryption.
- ▶ FHE: Fully Homomorphic Encryption.
- ▶ PoR: Proof-of-Retrievability.
- ▶ PoS: Proof-of-Storage.
- ▶ SMPC: Secure Multi Party Computation.
- ▶ TEE: Trusted Execution Environment.
- ▶ WASM: Web Assembly.
- ▶ ZK: Zero-Knowledge.