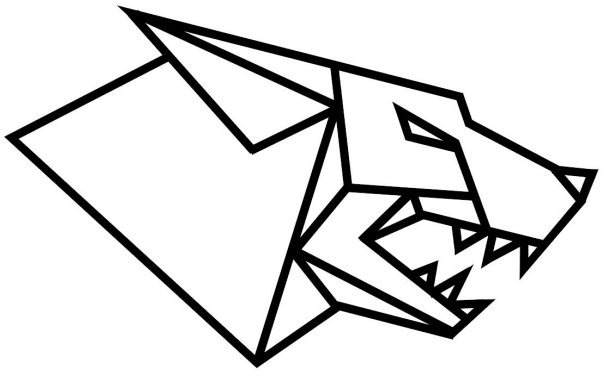
HellHound - BlackPaper

Mission statement document (not a technical paper)

Amira Bouguera (MS Applied Mathematics, MS Cryptography and Security),
Sajida Zouarhi (MS Engineering, Computer Science PhD student)



HELLHOUND

PRIVACY BY DESIGN FOR ALL (D)APPS

"If you want to keep a secret, you must also hide it from yourself."
— **George Orwell, 1984**

Table of Contents

1. Abstract

HellHound is a decentralized blind computation platform.

Blind computation is a process by which a function (or circuit) is successfully applied on encrypted inputs to provide the expected output. This output, depending on the use case, can be encrypted as well or it can be a plain text.
It is called a “Blind” computation since no one, except the data provider, know the actual data being computed.

HellHound’s goal is to provide a truly decentralized computing environment and a set of cryptographic tools to enable dapps developers to implement privacy-by-design.

Cryptography has been around for a long time and is used in several industries to secure exchanges and protect data. It has however often been out of reach for developers with no computer science or mathematical background.

The reality is that not all developers need these scientific backgrounds to build their great applications or services. Unfortunately for their users, implementing security and privacy takes time and skills which most early stage startups don't have. We also believe that blockchain startups of today might be the big players of tomorrow's web 3.0.

We think that by making the developer’s tasks easier, more startups and companies will adopt best practices in terms of privacy and foster the emergence of self-sovereignty for their users. We also strongly believe that providing a blind and verifiable computation platform, without the need of having an omniscient third part in the process, will unlock a large pool of untapped opportunities.

We have a chance at rebuilding the internet, so as Richard Branson put it "let's not screw it, let's just do it".

2. Introduction

a. Privacy by design for all dApps!

What is Privacy?

"The right to be let alone"

In 1890 (future U.S. Supreme Court Justices) Samuel Warren and Louis Brandeis defined privacy as the right to "personal autonomy, freedom of association, moments of reserve, solitude, intimacy, and independence". (ref)

More than a century later, we now have another term, "informational privacy," which means that all information about a person is in a fundamental way her own, for her to communicate or retain as she sees fit.
We consider privacy to revolve around **control**, **use** and **disclosure** of one's personally identifiable information.
Privacy is the cornerstone of many rights and freedoms that are important to us, and will always be a vital component of free and democratic societies.

However, our increasingly technologically-driven world puts great pressure on privacy. This is why designing an information technology (a system, a service, an application) without taking into account Privacy from the very beginning is dangerous and hostile to freedom.

<u>The Facebook/Cambridge Analytica example</u>

Facebook exposed data on up to 87 million Facebook users to a researcher who worked at Cambridge Analytica, which worked for the Trump campaign.
This amount of data was harvested through a Facebook app (a quiz).
It not only collected data from people who took the quiz but exploited a loophole in Facebook API that allowed it to collect data from the Facebook friends of the quiz takers as well. Facebook prohibited the selling of data collected with this method, but Cambridge Analytica sold the data anyway.

Privacy by design could have helped avoid such thing to happen at different stage of the process (data were sold, data were used but the intent wasn't disclosed to the users, data of friends of the user were used without consent etc.)
The Facebook/Cambridge Analytica scandal suggests that the EU has made the right choice to propose and carry out the GDPR (General Data Protection Regulation).

Applied since 25th May 2018, this regulation will bring several improvements to deal with data protection violations in the future.

Privacy by Design

Privacy by design is an approach to systems engineering initially developed by Dr Ann Cavoukian. The privacy by design framework was published in 2009 and adopted by the International Assembly of Privacy Commissioners and Data Protection Authorities in 2010. Privacy by design calls for privacy to be taken into account throughout the whole engineering process.
The European GDPR (General Data Protection Regulation) incorporates Privacy by Design. Business processes that handle personal data need to be designed and built in alignment with principles such as pseudonymization, full anonymization when appropriate, highest-possible privacy settings by default. Data is not available publicly without explicit, informed consent, and cannot be used to identify a subject without additional information stored separately. The data subject has the right to revoke this consent at any time.

b. A decentralized environment for trusted computation

Throughout history, technological advances have tremendously expanded trust through innovations such as the internet, secure e-commerce, social networks, exchange platforms for services etc.
Each of these innovations requires some trusted parties to engage together. However, these parties can be hacked, compromised or even dishonest. Blockchain technology has the potential to revolutionize the digital world by enabling a distributed consensus where each and every online transaction, past and present, involving digital assets, is trustless, practically immutable, and can be verified at any time in the future.
Ethereum blockchain offers more than peer-to-peer digital assets transfer. It has the capacity to perform any complex computation using fully trustless, quasi turing-complete smart contracts. Despite the great potential Ethereum has to offer, it still suffers from two major issues:

1. **Scalability**: Ethereum is not scalable due to the fact that every node in the network needs to process every transaction, validate it and store a copy of the entire state. The number of transactions Ethereum can process cannot exceed that of a single node which is currently 15 transactions per second due to the gas limit. Also transaction processing and verification speed is low to execute on any type of computation whether simple arithmetic or heavy computation.
2. **Privacy**: Information is published unencrypted to the public ledger and does not have any restrictions on access permission since everyone is able to read and process it.

There have been multiple solutions proposed to treat both of these problems such as sharding and off-chain computation. Both of these solutions intend to create a second layer of computation in order to reduce the load on the Ethereum network. Off-chain computing allows data to be processed off-chain and not published on the mainnet. However, the main issues for users are security and privacy since they interact with untrusted parties.
How to guarantee data confidentiality while being be able to perform any sort of computation on it? How to provide sound proofs that a computation was done correctly?
These are the main scientific challenges tackled by HellHound.

HellHound will enable an off-chain network of nodes to execute various types of computation over encrypted data and verifies the integrity of these calculations using different cryptographic tools.

c. State of the art - Actors of the Data Computation Ecosystem

Outsourced computation allows users with low computation power to delegate the heavy computation tasks to large data centers with more computation power. This delegation saves time and money for users while relying on a managed service platform. Ethereum blockchain suffers from scalability issues because smart contracts cannot perform large computation on-chain. The reason why smart contracts are constrained is the GasLimit, which is used to ensure the computation is bounded and it's a countermeasure for DOS attacks. Another issue is that the same computation are run by all the nodes of the network which is not optimized. Nor is it necessary if the trust in the computation doesn't rely on having a majority of honest nodes which is a requirement for most Blockchain protocols. Last but not least, we mention one of the biggest bottlenecks for scalability "Verifier's Dilemma". In Proof of Work models, miners get rewarded by solving a cryptographic puzzle. In order to solve this puzzle, miners spend computation power and verify the integrity of the transactions committed to the blockchain. The problem with this mechanism is that miners are not incentivized to participate in the verification process when they have a computational intensive transactions. This could allow a bad actor to compromise the integrity of the network by publishing invalid transactions. Another outcome could be that a bad actor would mislead miners to try and verify intensive transactions and exhaust their computational resources while it is mining the new blocks in the network.

In order to solve these scalability issues, several actors in the space have started developing off-chain solutions where they do all heavy computation off-chain and commit only results back on-chain. However, this raises two problems:

1. **Data privacy**: is data being shared publicly with those untrusted actors?
2. **Verified computation**: how do we guarantee that those actors were honest and delivered back correct results?

We will mention here some of the well-known actors in the blockchain space working on solving the problems of scalability, privacy and verification.

IExec, Enigma, Golem and Truebit are developing solutions to do off-chain computation in exchange for rewards. Most of them use hardware approaches (i.e secure enclaves) for this role and provide verification using different methods which will be discussed in the next paragraph.

On IExec, worker nodes use secure enclaves to conduct computation over data in plaintext (non-encrypted) and submit their results back to the user.

The user then compares the different results she receives and will accept only one based on the likelihood threshold between results using Sarmenta's voting mechanism (ref). This validation method relies on the fact that multiple nodes of the network will do the same computation task and send back their responses. If k out of n, $(k < n)$ number of results matches, then the result will be accepted and considered to be valid. This method however is easily attacked as it does not conduct an actual verification of the result correctness which leaves the risk of colluded malicious workers sending false results. Also, our research has shown that the hardware approach IExec is taking for computation is vulnerable to attacks (more on this in the following section).

Golem is using another verification approach called "**probabilistic verification**". (ref)

This verification method only calculates a fraction of the task, if this fraction is verified as correct then the system accepts the full result. The problem we see with this method is that for some stateful-dependent (ref) use cases such as 3D rendering (i.e. the use case that Golem is approaching), verifying a fraction of the task will not be enough to conclude whether the computation was done correctly or not.

TrueBit, on the other hand does not verify computation unless a dispute is raised about the integrity of a calculation by a worker. Dispute resolution occurs as an on-chain verification game where miners are the judges between a challenger and a solver (both are workers). The verification will only be done on the targeted part of the dispute but the workers needs to narrow it down by search through the merkle hash root of computation results.
This raises the following question: if any worker can raise a dispute, a malicious worker can launch a DOS attack on (or at least waste computation resources of) the network. The network of workers will spend time trying to solve the disputes which means searching through the merkle hash roots for several rounds. To mitigate this issue, Truebit uses staking with slashing conditions so that if a verifier triggers a false dispute to spam the network for example, the verifier loses its stake - if not, it is returned to the verifier.
However this ultimately doesn't prove correctness of the whole computation since incorrect results can stay unverified.

All of the previous mentioned solutions focus on solving the scalability problem and not the data privacy challenge.

Enigma is working on solving privacy issues. To do so, they partnered with Intel to use their SGX enclaves for computation over private data. (ref)

In the next section we will see that hardware approaches like Intel SGX are not enough to guarantee privacy preserving computation. In some cases, they can even leak sensitive data to attackers.

The following table compares the different approaches:

Decentralized Cloud computing projects	**TrueBit**	**Enigma**	**Golem**	**Iexec**	**HellHound**
Privacy Approach	Software based	Hardware based	Hardware based	Hardware based	Software based
Decentralized computing	Yes	Yes	Yes	Yes	Yes
Verifiable computing	Yes (1)	No	Yes (2)	Yes (3)	Yes
Recommendation engine	No	No	No	No	Yes
Blind computation	No	No	No	No	Yes
Decryption side	No use of encryption	Server (inside SGX)	No use of encryption	Server (inside SGX)	Client side

Fig X. Overview of the actors our our ecosystem

(1) challenge a computation by taking it to a court (miners are the judges) and there are a challenger and a solver. If the challenger questions the integrity of computation, the solver will be forced to play a verification game
(2) There is a portion of computed value to be verified
(3) The accepted result must reach the expected likelihood threshold of votes to reach consensus

There are other projects working on solving both privacy and scalability problems like Monero, Zcash and Starkware. The previous mentioned projects rely on advanced zero knowledge verifiable computing techniques like Bulletproofs, ZKSnarks and ZKStarks. These techniques allow a prover to convince a verifier about a certain claim by only sharing a proof which backs up the prover's claim without sharing any private information. These sort of techniques are usually used to send *confidential transactions (ref)*, where the transaction amount is hidden from public view in the blockchain. By including a zero-knowledge proof of validity in each of these transactions, network validators or miners can still verify transaction integrity without reading its amount.

The following table compares the previous techniques in terms of complexity, the need for a trusted setup and interactivity:

Protocol	Prover time	Verification time	Proof size	Trusted setup	Interactivity	Quantum resistance
ZKSnarks	Quasi-linear	Logarithmic		Yes	No	No
BulletProofs	Linear	Linear	Scales linearly with computing time	No	No	No
ZKStarks	Quasi-linear	Poly-logarithmic		No	No	Yes

- **Trusted setup**: It is a process performed by a trusted party or parties to generate a pair of keys to be used by the prover and verifier later on to generate and verify proofs.
  The trusted party uses a random secret as a seed to generate the key pairs, this seed is destroyed afterwards.
  The challenges regarding this process are as follow:
  1. Possibility to generate false proofs if prover gets access to the secret seed or if the trusted party is malicious.
  2. Not scalable: The trusted setup scales linearly with the computing time in ZKSnarks. This is way, the key and proof generation are done only once off-chain.

- **Quantum resistance**: ZKSnarks and Bulletproofs are not quantum resistant which means the security assumption behind each of these protocols could be broken eventually by quantum computers (see the quantum resistance section for more details).

- **Interactivity**: Prover and verifier exchange messages (challenges and responses) so that the verifier checks the validity of a certain proof sent by the prover. All the previous mentioned protocols can be made non-interactive by replacing the communicated messages by cryptographic random functions.

- **Performance**: ZKStarks is the most scalable solution between all of them and Starkware is currently using it as a scalability solution while Zcash for example is focusing more on the privacy aspect.

d. Why a Software-based approach?

We will define here both software and hardware approaches and their concepts. Then we will compare them in terms of privacy vulnerability to determine the best approach to consider.

1. Software: Homomorphic Encryption (HE) / Multiparty Computation (MPC)

Software based approaches leverage the usage of cryptographic algorithms such as HE and MPC while doing computation over data.
Homomorphic Encryption methods allow calculation and search to be made over encrypted data without the need to decrypt it.
Multiparty computation methods allow data to be split between multiple servers, where each of these servers apply computation on the data share they have. Data might not be encrypted but then will be unreadable unless there is a collusion between servers above a certain threshold.
More explanation about these protocols in the cryptosystems section.

1. Hardware: Secure Enclaves

Secure enclaves are hardware components which allow an isolated execution of code and data in an untrusted environment while protecting the areas of execution in memory. The access to application memory is restricted by hardware and remains secure even if the OS kernel is being hacked.
There are different technologies that provide secure enclaves solutions such as Intel SGX, AMD Secure Encrypted Virtualization, IBM SecureBlue++ .etc. In this paper, we will mostly focus our comparison on Software Extension or “Intel SGX” Trusted Execution Environments.

2. Why can Hardware approaches be dangerous?

Different decentralized cloud services such as Enigma and IExec have partnered with Intel to use Hardware solutions like Intel SGX in order to execute computation over sensitive data. In this computation context, data should be protected from any type of modification or access.

However, most of Intel Hardware chips have been found to be vulnerable to some attacks (Meltdown, Spectre and Foreshadow).

The first two attacks (Meltdown, Spectre) allow an attacker to access private data by misleading speculative scouts into a speculative execution attack. Speculative scouts's original mission is to improve performance by guessing values to complete future tasks. Compared to other Intel Hardware chips, the SGX is resilient to speculative attacks as it uses a security mechanism called Abort Page Semantics which denies writing attempts and always returns the same value for reading attempts from outside the enclave.

Unfortunately, SGX are vulnerable to another type of attack called Foreshadow. This vulnerability enables an attacker to create a shadow copy of the protected data into a different unprotected location. Then the speculative attacks are redirected to this new location and can now read the entire SGX enclave's memory content.

Another reason why hardware approaches are not privacy preserving is that data is encrypted on the client side with the public key of the server (which is an untrusted party), instead of being encrypted with the client public key. This system requires data within the enclave to be decrypted before being processed. If an attacker manage to access the enclave, it will then be easy to recover plain text data.

We will show in the following paragraph why it is considered to be a disadvantage.

3. Client vs Server side decryption

Data encryption is a very essential security measure when dealing with untrusted servers to perform computation over users' sensitive information.

There are two types of encryption depending on who owns the private key to decrypt data, whether it is the client or the server.

- **Homomorphic Encryption**

Homomorphic encryption allows data to be encrypted using the user's public key on the client side. Encrypted data is then sent to the server where computation will be performed. This method guarantees data privacy since information will never be disclosed to untrusted third parties, including HellHound nodes.

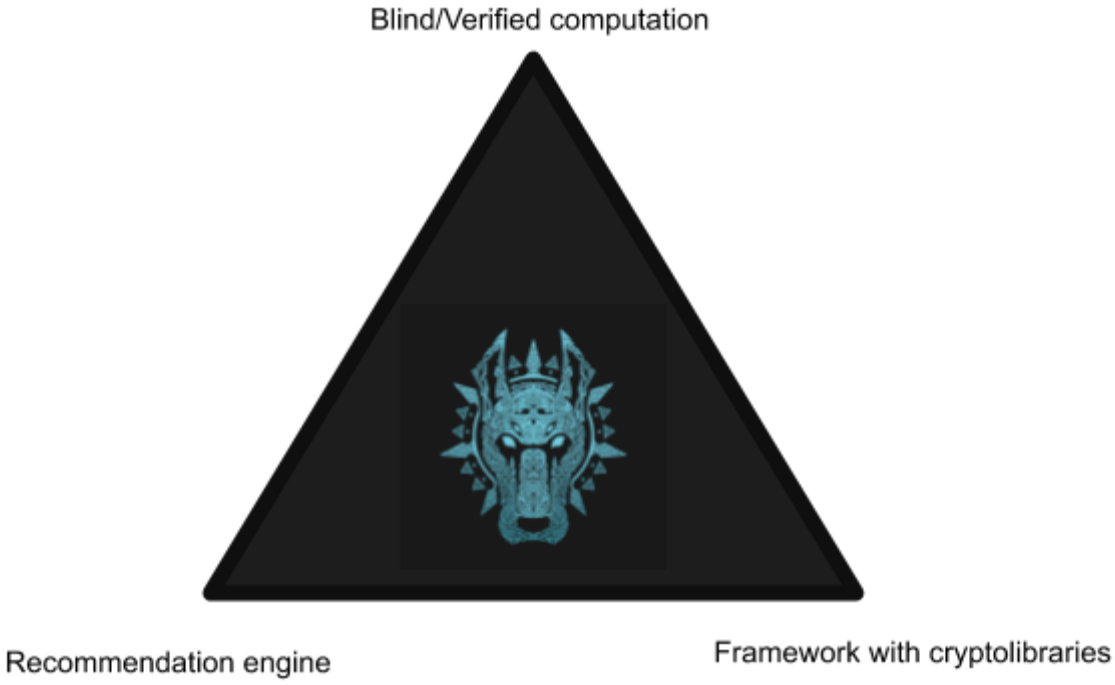
- **Memory Encryption**

With a secure enclave, data is encrypted using the enclave's public key, decrypted when it is inside the enclave and computation happens on plaintext data.
Then a Memory Encryption Engine is used to encrypt data on the way out of the enclave. This method represents a risk for privacy. With attacks like Spectre, Meltdown and Foreshadow, attackers will be able to access the sensitive data stored in the enclave and since it will not be encrypted, they will be able to both read and modify it. These attacks don't leave a trace, which means the client might never know it happened.

Both methods have their advantages and disadvantages, server side encryption will allow faster computation process than client side encryption but it does not protect data confidentiality nor integrity.
In this project, we leverage software cryptographic approaches to tackle privacy issues.
Our commitment is that we will never decrypt user's data.

3. Introducing the HellHound project

a. HellHound Triptych △



HellHound is a modular platform that provides 3 main components which constitutes the pillars of our triptych:

- a **recommendation engine** based on an expert system used for helping the user in her decision-making process on which cryptosystem to use;
- a **blind computation platform** enabling privacy as a decentralized service (PAADS) and **verifiable computation**;
- a **framework** composed of **verified crypto libraries** that will be in the future integrated with the favourite tools of dApp developers.

Each one of these components can be used independently.

i. Pillar 1: Recommendation engine overview

One of the three main components of HellHound is the recommendation engine, Pythia.
Pythia's role is to help the user select the best cryptosystem for her use case.
Depending on which inputs her dApp is processing and what computation she is seeking, the user is able to tap into the expert curated knowledge database of Pythia.
Pythia is a bridge between the user world and the HellHound world. It doesn't require any advanced mathematical or cryptographical knowledge.
There is a challenge in terms of ontology creation and search algorithm to provide the best recommendation. Our main challenge for Pythia will be the design of a user-friendly UX to make it simple to interact with.
To learn more about the Recommendation engine, see - Pythia section for more detail.

ii. Pillar 2: Privacy as a decentralized service (PAADS)

The core of the HellHound platform is the network of HellHound nodes. It can be viewed as a sidechain that aims to extend the current functionalities of the Ethereum main chain regarding privacy and scalability for heavy computation tasks.
This network of nodes communicates using the HOWL protocol (see section about HOWL for more details).
Their goal is to perform computation in a (truly) decentralized fashion.
Computation is not always duplicated on the different nodes; each HellHound node can work on a different set of instructions. HellHound nodes are not all mobilized for each computation, they work in a cluster (or "*pack")*.

"Privacy as a decentralized service" conveys the idea that any user (developer, company) can leverage our system without having to deploy a dedicated infrastructure.
We added the term "decentralized" to highlight the fact that this service is not owned by a central authority (e.g. HellHound).
This platform can be used by anyone to interact with the Hellhound nodes using tokens.

HellHound Tokens are part of the delivery mechanism for nodes' reward. Each HellHound node receives upon completion of its task an amount of tokens that is calculated based on the heaviness of the computation it just performed. We will not discuss a Token model in this paper but the idea is pretty straight forward if we compare it to Ethereum computation formula.

Our client is named Cerberus. Cerberus will provide a UI enabling the user to monitor any deployed services and to manage her token balance.
It can be accessed on a web app or it can run on the user's desktop as a light-client depending on the features embedded (e.g. key management).

HellHound platform provides a suite of smart contracts known as the STYX. Their main role is to anchor the critical elements (proofs) that the user, the hellhound nodes, or anyone else, might need to prove a computation.
The STYX also manages the reward distribution in a decentralized manner as well as the deposits (staking) needed in the platform. We provide a decentralized reputation system that will also be anchored in the STYX to express the current reputation of the HellHound nodes.

To learn more about the HellHound nodes, Cerberus, Styx and the Howl protocol, see the respective sections.

comment

iii. Pillar 3: Framework and libraries

Our goal is to make the task of implementing privacy by design as easy as possible for developers.
This is why we are going to propose **open source libraries and smart contracts** for the community.
Developers will be able to use them to start implementing privacy by design in their dApps.

1. We are currently working with other projects in the blockchain space to integrate general purpose HellHound smart contracts in XXXX.
2. We also plan to release Cryptographic Libraries that other protocols or projects can use safely. Developers of traditional Apps or dApps will be able to leverage this list of certified crypto libraries in their architecture with no technical overhead.

Making available to the community verified cryptographic libraries

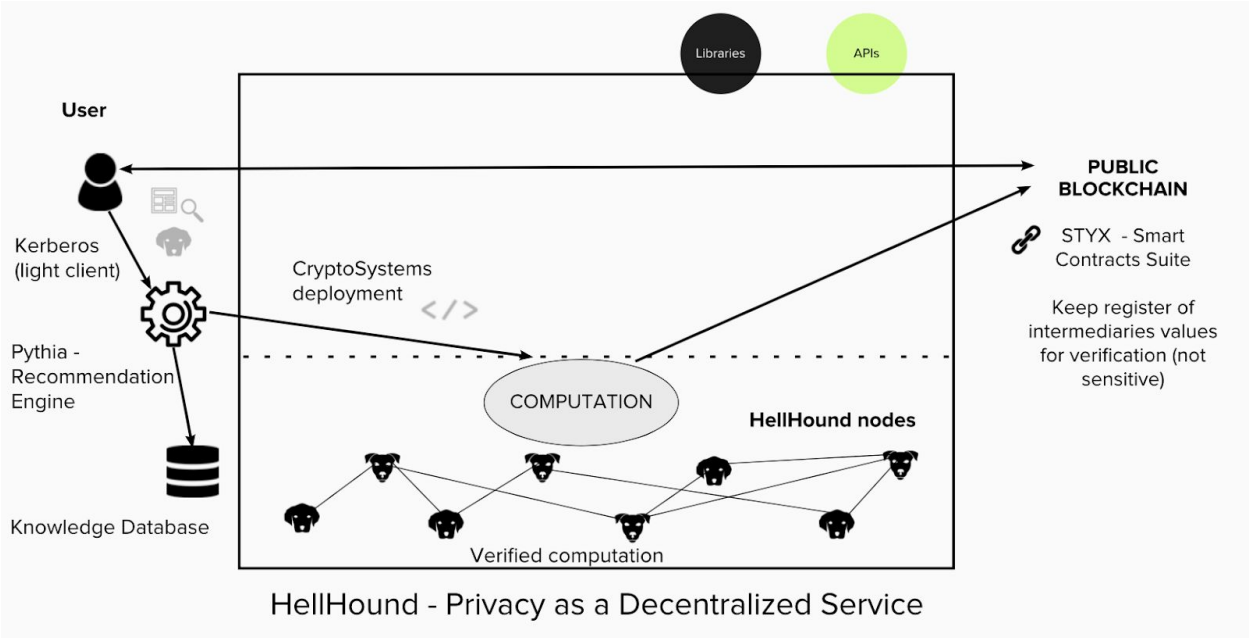
Cryptographic libraries are used as tools to implement security and privacy measures into applications. The fact that application developers rely on these libraries brings the need to make

sure that they are not vulnerable and exploitable by hackers. In the last few years, many widely used libraries have suffered from critical flaws.
For example, the OpenSSL library has suffered from different types of bugs affecting the TLS protocol layer, such as the HeartBleed attack, as well CacheBleed vulnerability which was a side channel attack against RSA implementation and many others over the years.
The Libsnark library used to implement ZK-SNARKS protocol has also suffered from a vulnerability affecting the soundness property of the protocol. This bug allowed the protocol's implementation in libsnark to accept false proofs, such a vulnerability could have affected the whole verification system Zcash is based on.

In order to find bugs in the cryptographic code, there have been different methods used like testing, fuzzing, formal verification etc. Formal verification proves or disprove mathematically the correctness of a protocol implementation as a whole.

We aim to provide a curated registry of verified secure cryptographic libraries to the community in order to minimize the exposure of dApps to these types of attacks. This work will be done by HellHound in collaboration with researchers from the Formal Verification space.

b. HellHound Architecture Overview



HellHound - Privacy as a Decentralized Service

c. Pythia - Recommendation Engine

Our approach for the Pythia Recommendation Engine is to model an ontology that will:

- contain the logic and knowledge needed by the recommendation engine;
- provide a construction algorithm that will create combinations under constraint by finding a path in the graph;
- compute the corresponding QoS (quality of service);
- create an answer for the UI component of Cerberus for an intuitive visualization of the results.

The recommendation engine is constituted of two main components :

- database containing rules and facts (ontology, graph)
- combination algorithm for the decision making

The knowledge database contains the current state of the art for HE, SMPC and VC (verified computation). It also contains the logic rules based on criterias explicited in the CryptoMap (a document we are working on at the moment and that we will release for the community).

Based on that, Pythia decides which CryptoSystems, Settings to select in order to match the Client request with the best QoS possible.

Query
The user fills the form presented on the interface of our website. Her query is sent to Pythia to be analyzed.

(put here the serie of questions we used in the demo or a screenshot if quality isn't bad)

Pythia takes these parameters as input.

Using the knowledge database and the decision algorithm, recommendation of CryptoSystems are generated. This step is always free and can be done by the user as many times as she wants.

Analyze
In this step, we assume that the user has an algorithm ready and wants to use Hellhound to run it, following the recommendation Pythia just provided. The user needs to stake to access the following step.
Once done, the user will send her algorithm code on the platform for analysis.
Cost computation is estimated in order to determine the price of the computation that the user will have to pay (hellhound nodes reward).

Validate
If the user agrees on the price, the HellHound nodes selection starts (see HellHound node selection for more information). A pack is formed.
The user, through her client Cerberus, sends a transaction of “initialization” to the blockchain with the cost of the computation (the Styx will keep it safe in an escrow and will only pay the HH nodes involved after completion of their task).
This will also trigger the deployment of the virtual machines (or containers) on the selected HellHound nodes. The Pack is ready to start. The next step will be to use Cerberus to encrypt locally the user’s data and then send it directly to the selected HellHound Nodes.

The Service is deployed on HellHound
Each HellHound node runs a virtual machine, just like the EVM on Ethereum.
Using a dockerized approach, each new cryptosystem (previously selected by Pythia and approved by the user) will trigger the creation of new containers embedding the cryptographic set up needed for computation and ready to receive the information.
After a computation is completed and all proofs are saved, the HH node can destroy its containers and move on to the next job in its queue.

d. HellHound nodes

i. HellHound clients and the Howl protocol

Cerberus client and HellHound nodes have in common these components:

- they have a Virtual machine,
- they maintain a Local storage (file system, reputation matrix),
- they use the HOWL protocol to communicate.

Any user can download a Cerberus client for free. They can use it as such (to perform computation on HellHound) or they can become a HellHound node. Being certified as a HellHound node requires going through a vetting process.
Installing Cerberus isn't needed to interact with Pythia. She will be available online as a free service. However installing Cerberus is required if the user select a multi-party computation approach, Cerberus becomes one of the parties involved in the computation alongside HellHound nodes. This is why a Cerberus client and a HellHound client have so much in common.

Main attributes of Cerberus Clients or HellHound nodes:

- Available memory
- Available processing power
- Geographical location
- Reputation matrix

One needs a **certified client** to become a HellHound node. Certification is delivered upon completion of the *Purgatory* vetting process.

	Cerberus	HellHound
Local storage (> xx mo)	yes	yes
Computation power (> %)	yes	yes
Reputation matrix	yes	yes
Certificate	no	yes
Human Interface	yes	no
HOWL protocol	yes	yes

All HellHound clients (Cerberus, HellHound nodes) share the same communication protocol: the **HOWL**.
The HOWL protocol defines all communications happening between cerberus nodes from the authentication of a new client (Cerberus trying to connect to the HellHound platform) to the P2P auditing mechanisms between HellHound nodes (see Howl's decentralized reputation system).

ii. HH Node Purgatory

Only nodes who have successfully completed *Purgatory* (vetting process) can register to become a HellHound.
Nodes which want to join the network as dedicated computing resources will apply on-chain to the STYX contract. This application mechanism allows Hellhound to have a permanent transparent record of the participating nodes.

Purgatory steps:

I. **Requirements**: computing power, resources, legal commitment, SLA etc.
II. **Private key storage:** the private key of each node must be protected and stay safe (e.g. in a cold storage such as Ledger).
III. **Identity verification:** The STYX contract will verify the identity of each node on the public blockchain with some information related to the node owner IRL since the participation is not anonymous and there is a legal framework around this participation.
IV. **Staking**: Each node which want to join the network as a computing service must put an amount of HellHound tokens at stake. Doing so can be made easy since HellHound accepts other cryptocurrency and even fiat, but everything is converted to HH tokens in the Smart contract. The stake is a way for the nodes to prove their commitment and to make sure they act honestly during computation/verification rounds. The security deposit will be locked in an escrow smart contract [1]. The value of the security deposit will be

set according to the risk parameters of the node (unfriendly legal framework in the country of the node will increase the stake). It also depend if the node wants to run very heavy computations (more costly so more lucrative), if yes the stake will be higher (more to gain if honest, more to lose if not).

V. **Confirmation**: After all requirements are verified and the nodes submit their stake, HellHound contract will confirm their registration to the network both legally and on-chain.

The result of this process is the issuance of a **Certificate**.
We are still reviewing the details of the purgatory process and the attributes needed to generate the Certificate. We envision a process both automated and human were capacities of the nodes are declared (on-chain), verified by a team of auditors and then once the nodes are in the network, the HOWL protocol will help us monitor their activity.
Any malicious behavior will lead to a warning and eventually to the loss of their stake based on the indicators provided in a decentralized way by the network and if need be an audit.

Certificate Issuance:
It's not centralized because the vetting process will be transparent (stake amount will be computed automatically etc), so it's not like we can just decide to say no. But at the same time, if something seems risky, we act as a high entry barrier to HellHound. Our model isn't to be an open network nor to have anonymous node. Every node is known, there are legal ties because why not use that as well if possible, and entering the network isn't easy by design. But difficulty here doesn't mean centralization, because it comes with a transparent process that anyone can try and some might fail. We are looking into "attributes certificates" that will be generated based on the capacity of those nodes. They can't cheat those and neither can we."

Example: If Google wants to dedicate computing resources to HellHound, they can download a client first, generate a pair of ECC keys to be used as their identity and register on-chain to the Styx.
They will then have to sign an e-contract stating that they commit to provide at least a certain amount of computation and availability and go through the staking phase.
Once everything has been verified, a unique certificate is delivered to their client, turning it into a fully functional HellHound node.

iii. HH Node Selection

For each computation round, a subset k-of-n nodes will be chosen to perform this computation. The selection is divided into two steps:

1. **Pre-selection:**
   During this phase, a subset m-of-n nodes will be selected based on different factors such as

a. computation capacity of the nodes: computation power, available memory, geographical location etc.,
b. reputation:
    i. HOWL's Decentralized reputation system.
    ii. HOWL's Continuous audit interaction between nodes (challenges).

2. **Random selection:**
   The second round of selection will be taken only from the subset m and not the whole network since only these nodes are eligible for the work according to their capabilities and reputation. HH runs a random function selection to determine the identities of the nodes which will participate in the next round. Only a set of k nodes will be selected (the value of k is dependent on the cryptographic system used in the computation).

iv. HellHound Node participation

The chosen nodes will be notified when there is a request for computation, the request will contain the proposed price for each operation which was computed. The previous mentioned price will be computed according to HH standards (open formula) which nodes approve on when going through *Purgatory*.
Based on the SLA (service level agreement) they signed on-chain, HellHound nodes that are unavailable to perform a computation or that try to send back fake results will see their reputation decreased and their stake endangered.

v. HOWL

The HOWL protocol is used by Cerberus/HellHounds to communicate safely and in the most decentralized way possible.

The HOWL protocol is used for:

- Computing in a decentralized fashion reputation of the HellHound nodes,
- Communicating with on-chain elements (STYX smart contracts),
- Auditing the nodes to maintain a high-level of integrity in calculations using challenge games between HH nodes.

In this part we will introduce the Decentralized Reputation System of HellHound.

HOWL's decentralized reputation system
Hellhound uses a decentralized reputation system in order to calculate the aggregation of trust between the different Hellhound nodes. Each node knows its partial trust on its neighbors and wants to compute its trust on other nodes in the network. Considering that the trust of each

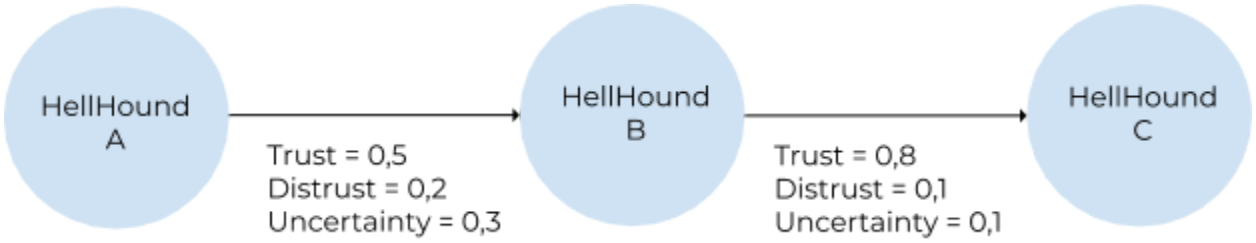
node for its neighbors is private, the reputation function should not reveal partial trust nor the final trust computed in order to guarantee both safety and privacy.

1. Step 1: Partial Trust

Nodes can create small groups of neighbors, inside each group, nodes challenge one another using different testing techniques to figure out which members of the group are up or down and which are honest or malicious. These challenges will allow every node to construct trust knowledge about its neighbors.
Trust is represented by a triplet (**trust, distrust, uncertainty**)

a. Trust represents the percentage of positive experiences,
b. Distrust the percentage of negative experiences,
c. Uncertainty would represent lack of information about another node (e.g. when a node is freshly added to the network). *uncertainty = 1 - trust - distrust*

Schéma 1



2. Step 2: Global Trust

The HH network contains N nodes of workers, each one of these workers has a partial trust on some of their neighbors but does not know the level of trust they could put on other workers. In order to compute the global trust, all nodes must collaborate and share their collective partial trust values encrypted with everyone on the network.

3. Step 3: Aggregated Trust

The aggregated trust is computed from the average (for each x, y in the matrix) of all global trust matrix computed during the previous step. This aggregated matrix of averages is stored on the STYX and is considered the absolute value of nodes' reputation (ref).

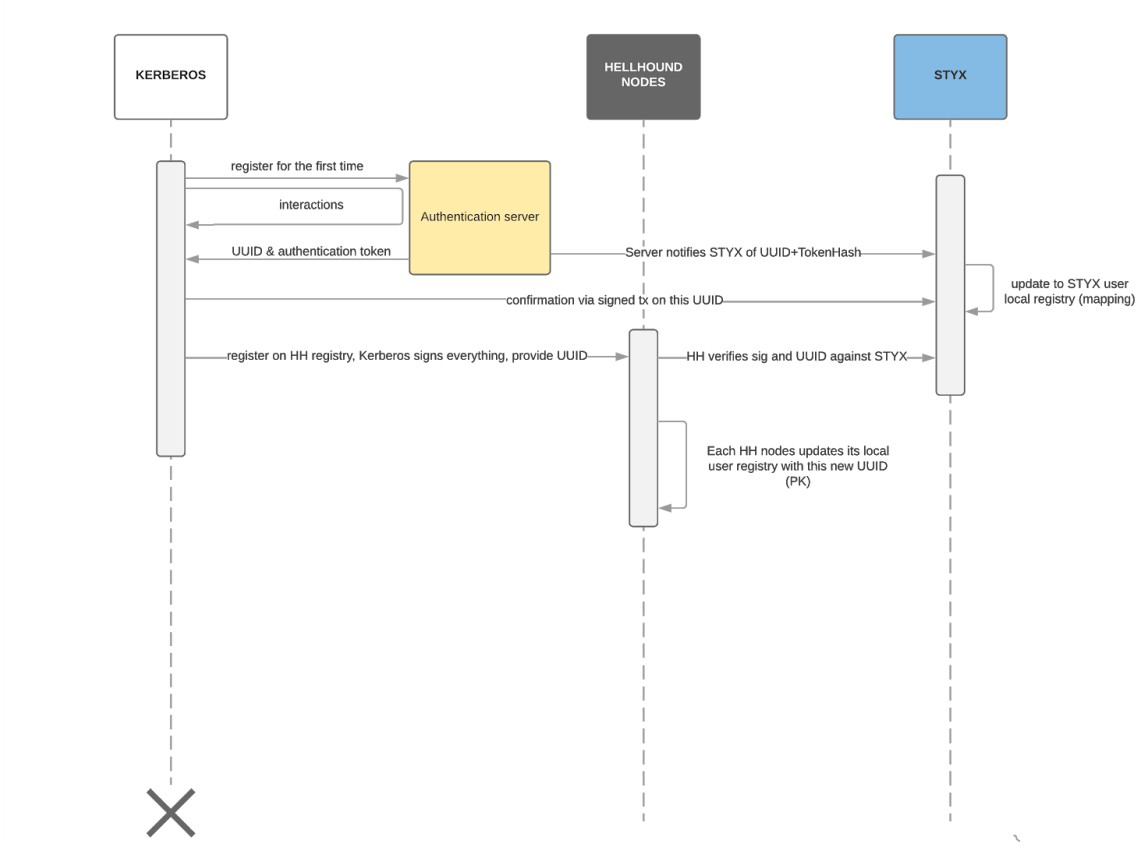
For drawing the figure:
Each node knows a line of the matrix, its line is at first incomplete and then it is completed through transitivity.
With N nodes we end up with a total matrix of NxN.
Then we can compute the average reputation by taking the values of each column of this Matrix.
We end up with N values of reputation for N nodes.

Hellhound could be one node on the network and have its own line of reputation. That hellhound node will contain at first 100% of uncertainty and will be completed after the MPC computation by the aggregated reputation values provided by the partial aggregated insights of the others nodes.

e. Cerberus - Light client for the user

Cerberus is a client with features that vary from those of the HellHound nodes.
Cerberus runs on the local environment of the user and acts as a bridge between the end-user and the HellHound platform.
This client is identified during the **authentication phase** with the HellHound nodes and with the Styx smart contract that handles an anonymous registry of all users. The authentication token that is generated during this registration phase enables Cerberus to interact securely and independently with the HellHound nodes.

The HellHound nodes can at any time search the blockchain to verify if the Cerberus client trying to interact with them is registered on Styx and that both identifiers (Cerberus client and the one on Styx) are a match. All transactions between Cerberus and HellHound or Styx are signed by Cerberus and verified.



Cerberus provides a user interface (dashboard) enabling the user to monitor the services deployed on HellHound, for example:

- the progress of her computation,
- the state of the Pack (HH nodes selected as the cluster to work on her task),
- the results (when available)
- and a verification option (involving staking for the user).

Cerberus can play the role of “Verifier” and take part in verification of computation by calculating pre-processed data that will be used after to verify the correctness of the HellHound nodes results, they act as “Provers”.
See “verification of computation with HellHound” and pseudo-seq diag III for more details.

f. Styx - Smart Contracts suite

STYX smart contracts handle the following aspects:

- Mapping of all Cerberus clients that wish to interact together
- Proofs generated during computation,
- Reputation (aggregated trust matrix),
- Security deposit of the Cerberus clients,
- Reward distribution for the HellHound nodes,
- Root hashes of computation steps (e.g. TrueBit)

Client and HellHound nodes can call the Smart contract to update intermediary values (proofs) needed during the MPC phase for example - *see Verification of computation on HellHound.* Each computation/query is identified on hellhound by its computation ID and linked to the address of the client (Ethereum or other blockchain address that has initiated the query). Requirement is that the same Ethereum address that was generated at the beginning for the Cerberus client at the “query & deployment phase” is the one used after for all interactions.

4. When and How to use HellHound?

a. Using HellHound for your products and services

Various use cases can be build using HellHound but to leverage this technology properly the use case need to be aligned with these assumptions:

- **User's privacy is a core value and responsible data management is a goal.**
- **Traditional actor handling the computation isn't trusted.**
    - If this is not case, then there is no need for a decentralized computation system since the choice of trusting a central authority has been made. This means that any company can run a blind computation service as a sole entity and the customers will still need to trust it for the correctness of the result.
- **The time needed to verify the computation is shorter than the time needed to perform the computation.**
    - If this is not the case, then any actor can just do the calculation itself. There is no need to delegate the computation and no need to worry about a potential disclosure of information to a third party. Of course, if the use case involves heavy computation (compute-intensive) tasks, then the computation need to be delegated.
- **The data needed to perform the computation come from multiple actors which don't necessarily trust each other.**
    - If this is not the case, then we would have one actor being able to encrypt all data with one key and delegate the computation to a 3rd part. This is very convenient but it doesn't work if you are processing data, for example in a matching service, coming from differents sources. Each sources will encrypt its data with its own pair of key, making it impossible for one cryptosystems to process the information. In that case, you will need a platform with a cluster of nodes able to perform SMPC over this computation and obviously those nodes need to be independent so it needs to be nodes coming from a decentralized architecture and node from one single company.

Encrypted data is stored on the hellhound nodes for the duration of the computation task

b. User story

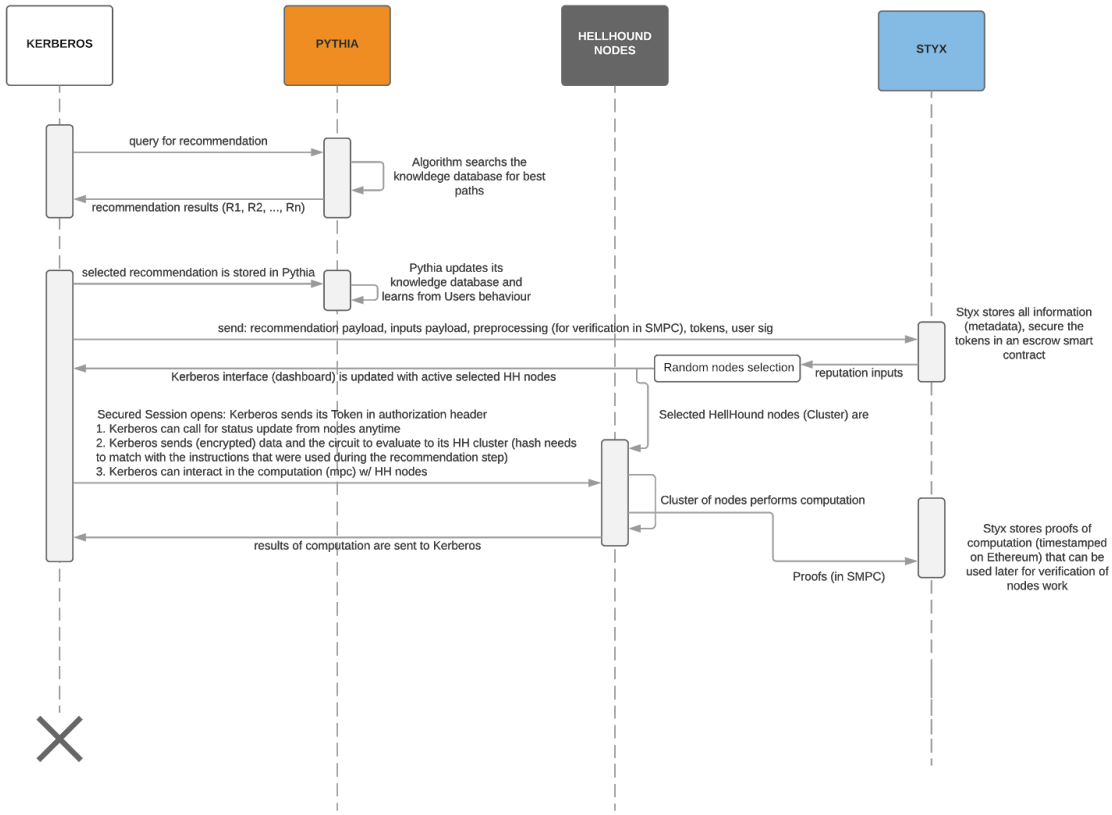
i. Using HellHound to know what Cryptosystems to use in your service

ii. Using HellHound to deploy your service

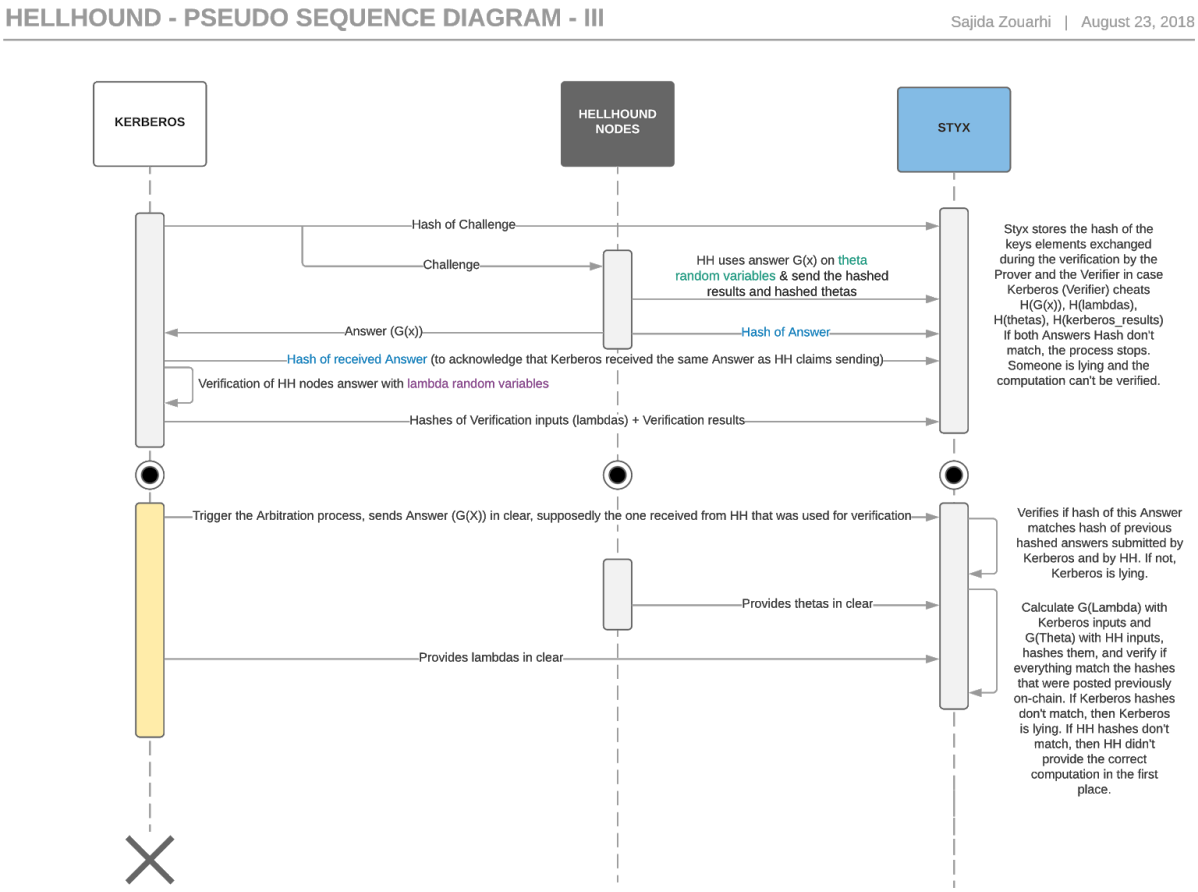
iii. Storage of encrypted Data

iv. Computation of encrypted data

v. Results can be encrypted or plaintext


KERBEROS
PYTHIA
HELLHOUND NODES
STYX
query for recommendation
Algorithm searchs the knowledge database for best paths
recommendation results (R1, R2, ..., Rn)
selected recommendation is stored in Pythia
Pythia updates its knowledge database and learns from Users behaviour
send: recommendation payload, inputs payload, preprocessing (for verification in SMPC), tokens, user sig
Styx stores all information (metadata), secure the tokens in an escrow smart contract
Random nodes selection
reputation inputs
Kerberos interface (dashboard) is updated with active selected HH nodes
Selected HellHound nodes (Cluster) are
Secured Session opens: Kerberos sends its Token in authorization header
1. Kerberos can call for status update from nodes anytime
2. Kerberos sends (encrypted) data and the circuit to evaluate to its HH cluster (hash needs to match with the instructions that were used during the recommendation step)
3. Kerberos can interact in the computation (mpc) w/ HH nodes
Cluster of nodes performs computation
results of computation are sent to Kerberos
Proofs (in SMPC)
Styx stores proofs of computation (timestamped on Ethereum) that can be used later for verification of nodes work

c. Verification of computation on HellHound



5. HellHound Use cases

The following use cases list are examples of services that can use HellHound to solve privacy and computation issues. We provide this list as an illustration of how HellHound adds value to dapps. Some of these services are currently being developed by different teams around the world.
HellHound's goal is to help developers and project owners understand how to leverage the HellHound platform.

MERGE THIS PART IN RED WITH THE USE CASE:

HellHound will provide different computation methods depending on the use case, calculations, inputs, time constraints etc. Our current approach focuses mainly on these families: SMPC, HE and ZKP. This part will provide some general information about these cryptosystems. Each cryptosystem has its pros and cons and it's hard to grasp them without a deep understanding of the maths behind. This is why we are writing for the community a "CryptoMap", a paper mapping the state of the art of the existing cryptosystems through several criterias such as time, complexity, interactivity, quantuum resistance etc.

i. SMPC - Secure multi party computation

A cryptographic method that allows n parties to jointly compute an agreed function f of their inputs $x_1, x_2, ..., x_n$ where each party i knows x_i.
Parties will compute $f(x_1, x_2, ..., x_n) = (y_1, y_2, ..., y_n)$ where each party i can only learn the output share y_i and no additional information.

ii. HE - homomorphic encryption

An encryption scheme E that allows computation on ciphertexts $c \in C$ (where C denotes the ciphertext space). , generating an encrypted result which, when decrypted, matches the result of the operations as if they had been performed on the plaintext.

RSA example: Multiplicative property

$$E(M_1) \times E(M_2) = M_1^{\sim e} \times M_2^{\sim e} = E(M_1 \times M_2)$$

Where M_1, M_2 are two plaintexts, E is the encryption algorithm and e is the RSA public exponent.

iii. ZKP - zero knowledge proof

A Zero knowledge proof is a cryptographic protocol $\langle P, V \rangle$, where a prover P can convince a verifier V that she knows a secret s without sharing the secret itself. The verifier responds to challenges set by the verifier by sending a proof of knowledge of s.

b. Key custody - key management

Key custody is a very important and sensitive topic that concerns cryptocurrencies. All current solutions that provide key protection are mostly hardware based or rely on central servers storage. Threshold cryptography and Homomorphic Encryption can be used as methods to guarantee a high level of security by distributing different key shares between several custodians who do not share any information between one another about their shares and manage still to sign transactions in a secure way.

c. DNA sequence searching

Example: A decentralized "23andMe"
A startup has developed a great algorithm that can, given health data from its user (here DNA), provide insights about origins and family connection to the user. However that startup wants nothing to do with the actual DNA of its users and believe they shouldn't have a database with this information under its responsibility. With HellHound they can create a service using blind computation and tell its users: "you only have to send an encrypted file with your DNA and you will get back your results, we will never see the file you send us and we will never see the results provided by our algorithm."
People here will pay to use their service (the unique algorithm) but they won't pay the "double price" of giving away their DNA information too.

d. Salaries - analytics without disclosing salaries

A tool that can take salaries and other information about employees and use a comparison algorithm to generate analytics.
To get some salaries data (in a private manner) we could use a zero knowledge proof system and figure out if gap differences exist between people (along very specific criterias such as gender, experience, location etc.) without disclosing anyone salary/name.
This way if gaps exist, the employees will know about it and won’t need to trust a central part (the HR service or the company in general) for a fair assessment of their situation.
Companies can have a fair and transparent process around HR & compensation policy without exposing themselves to legal risks or making some people uncomfortable within their organization with a 100% transparent policy.

e. Kidner - matching algorithm

Kidner is a platform that helps healthcare facilities finding a match for kidney paired-donation.
It's a global database, Blockchain-based, secured with cryptographic tools.
Kidner increases the chance to find rapidly a proper match while being fully protected.
Privacy and confidentiality are preserved thanks to encryption tools and certificates are issued by authorized healthcare entities. The algorithm operates in real time and is decentralized.

When a match is found, Kidner is notified and the recipient Doctor and healthcare professionals receive all the information needed to arrange the operation logistics.
Kidner offers the opportunity to extend paired exchange for all incompatible donor-recipient pairs in a secure and rapid way.
www.kidner-project.com

f. Enabling quantum safe approaches

The security behind most of the cryptographic protocols is based on the hardness of solving some mathematical problems. We take as example RSA (Factorisation problem) and ECDSA (Discrete Logarithm problem).

Blockchain technology relies on several cryptographic techniques in several ways:

- Public Key Encryption: sign transactions using digital signature (ECDSA algorithm) and for data encryption.
- Hash Functions: each block contains the hash (cryptographic fingerprint) of the previous block, these one way functions are resilient to collision which makes a single change on the input changes the whole blockchain history which is nearly impossible since attackers must control over 51% of the network to accomplish such attack.
- Merkle tree proofs: used to verify the sequence number of a specific blockchain transaction in the entire history of the blockchain.

Quantum algorithms like Shor and Grover can solve several mathematical problems. Shor for example can solve both the factorisation and Discrete logarithms in polynomial time which would break both RSA and ECDSA algorithms.

The goal of **post-quantum cryptography** (also known as quantum-resistant cryptography) is to develop cryptographic systems that are secure against both quantum and conventional computers and can interoperate with existing communication protocols and networks.
HellHound gives a set of different post-quantum cryptographic libraries to be used by developers while building their dApps in order to make sure that they will not be vulnerable to quantum attacks in the future.

6. Token model

a. Business model

Everytime the HellHound PAADS platform is used (i.e. an end-user sends token to STYX escrow contracts to deploy its circuits on HH, or for verification tasks), a small fee is sent to the HH treasure safe.

Add diagram

This will help HellHound adopt a sustainable model in the long-term and maintain the platform as well as multiple free services and tools by paying a team of researchers (cryptographers, computer scientist), engineers, blockchain architects, data scientists, full stack developers and cybersecurity experts.

It will also be used to set up **regular bounties** either to stimulate the ecosystem around complex issues or to incentivize people to attack the system and provide feedback to the team.
We are waiting for projects like Aragon or other to provide a structure for a true DAO, which our team is interested to adopt (decentralized accountancy of HellHound).
But for now, we will only use a structure that we know in order to minimize the risk by reducing the unknown variable. You can expect a core team to be fully (but transparently) in charge of HellHound at least for the first main milestones (HH testnet, HH mainnet, HH production, HH massive auditing and attack period) that might take around 2-3 years.

b. Incentivization system

i. Node incentivization

HH nodes must be incentivized to stay honest and deliver correct results, this will be guaranteed by applying a reward/punishment system that rewards the honest nodes and punishes the malicious ones.
Reward/Punishment: After the verification process is done, the security deposit will be given back to honest nodes and taken away from malicious ones to be distributed among the nodes which participated in the computation.

Decentralized reward delivery mechanism

The Styx contract keeps a record of the honest/dishonest nodes IDs in order to punish/reward them after finishing a certain computation/verification process. The steps are as follow:

- Styx recovers the stakes back from malicious nodes.
- Then, Styx adds it to the reward amount which was already paid by the user and kept locked with an escrow contract.
- Finally, Styx will distributes the overall amount between honest nodes in the current computation round.

ii. Cerberus incentivization

Cerberus clients must stake if they want to engage in a verification process with the HellHound nodes.

STYX contract makes sure both parties play honestly and do not cheat. So in case, the Cerberus client says he received a faulty result, there would be a judgement to determine if that was true or not, if not Cerberus will lose his stake.

7. HellHound attacks surface

There are different attack vectors which could threaten the security of HellHound platform, in the following section we mention these attacks and the mitigation methods used by HellHound to resist them:

A. DOS attack against Pythia (Pythia will be a centralized at first): set up of IDS and IPS (intrusion detection systems and intrusion prevention systems) those components analyse behavior of incoming requests to determine if they are malicious or not
B. Sybil attack
C. Zombie nodes
D. Malicious Cerberus
E. Cryptographic vulnerabilities: We will use formal verification methods to insure correctness of executed code. Formal verification methods will guarantee memory correctness, functional correctness, protects against side channel attacks....
F. Hardware vulnerabilities: we do not rely on hardware to preserve privacy at the core. We can use it as an added layer of difficulty for an attacker.

8. Join us in the HellHound journey

Sajida Zouarhi - Co-Founder - Blockchain architect

Sajida Zouarhi is a computer-science Engineer and works as a Blockchain Architect @Consensys. She is also a Computer Science researcher and worked with a major Telecommunications Operator (Orange) during her PHD thesis. Her research work focused on *private data transmission over communication channels on complex systems* and *quality of service on an end-to-end critical service* (recommendation engine, simulation).
She is President of the eHealth and Blockchain Think Tank and has been a technical advisor on the board of several Blockchain projects. As the founder of the Kidner Project, she collaborated with WHO to help prevent Kidney Trafficking using Blockchain technology. She is an

international Hackathon Mentor and organizer with the Blockfest that she co-founded in 2016 in France.

Amira Bouguera - Co-Founder - Cryptographer

Amira Bouguera is a Cryptographer, Mathematician and works as a cryptographer @Consensys. She has previously worked in the blockchain space prior to joining Consensys as a security engineer at Stratumn (traceability blockchain startup) and smart contract security auditor. Amira became interested in privacy and security related issues when she was studying cryptography. During that period, she worked in LJK applied math laboratory with one of the best international cryptographers on outsourced computation using Homomorphic encryption and Zero knowledge methods. She helped organizing the Grehack security conference and workshops. She is also an international speaker and gave talks about blockchain introduction, smart contracts security and cryptography during conferences (Dappcon) and meetups.

Here are some ways you can participate to the HellHound Journey.

9. Conclusion

In this paper we introduced HellHound, a decentralized blind computation platform.
We described the ecosystem of actors as well as the unique approach we chose to solve the scalability and privacy issues around data computation.
HellHound added value relies on a recommendation engine, a decentralized network of vetted nodes and set of tools (framework, crypto libraries) that will be easy and safe to use by dApp developers.
We believe that our solution will make dApps focus on solving the technical challenges they are facing in their applications and not having to worry about privacy of critical data being disclosed or tampered with.

Our vision is to make cryptography easily accessible to people with no cryptographic background in order to have more applications implementing privacy by design. It is our belief that while blockchains and decentralized technologies are laying a new path for the internet as we know it, we should focus on issues like privacy to create a better web.
We acknowledge the effort made by other projects working on privacy in the blockchain space and we hope to collaborate and find synergies that will put us closer to our goal: Privacy by design for all dApps.

10. Acknowledgments

HellHound is a ConsenSys formation that was initially supported by the Ethcompute applied research effort. We would like to thank Prof. Dumas from Grenoble Alpes University and LJK Lab for his contribution to the Cryptosystems research.

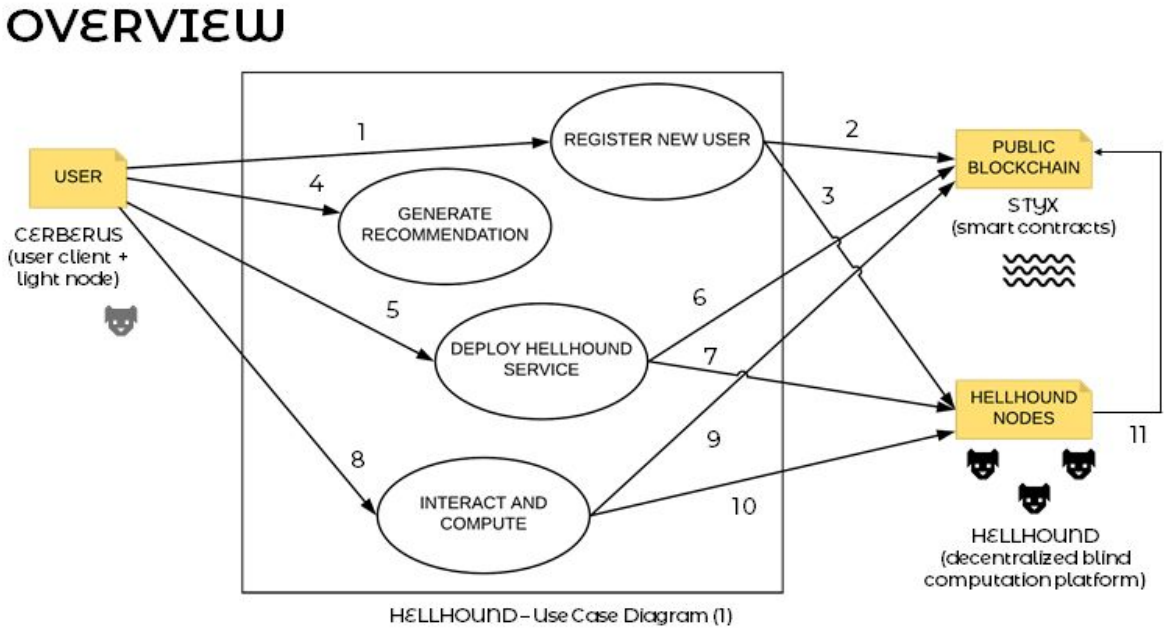
11. References

1. Source: Privacy by Design in Law, Policy and Practice, Dr Ann Cavoukian A White Paper for Regulators, Decision-makers and Policy-makers
2. Quantum computing and Shor Algorithm
3. Golem verification method for rendering
4. TrueBit verification game
5. Verified crypto libraries
6. Bug in libsnark
7. CasheBleed Side Channel Attack against OpenSSL
8. Multiparty computation
9. Verifier Dilemma
10. Proof of contribution (Iexec)
11. Verification game (TrueBit)
12. TrueBit paper
13. Stateful dependent (Golem)
14. Golem whitepaper (probabilistic verification)
15. Bulletproof
16. Monero Bulletproof
17. ZKstarks a scalability solution

18. SGX
19. Enigma - Intel collaboration
20. AMD Secure Encrypted Virtualization
21. IBM SecureBlue++
22. Meltdown and Spectre attacks
23. Foreshadow
24. Memory encryption Engine
25. Decentralized reputation system

Appendix A

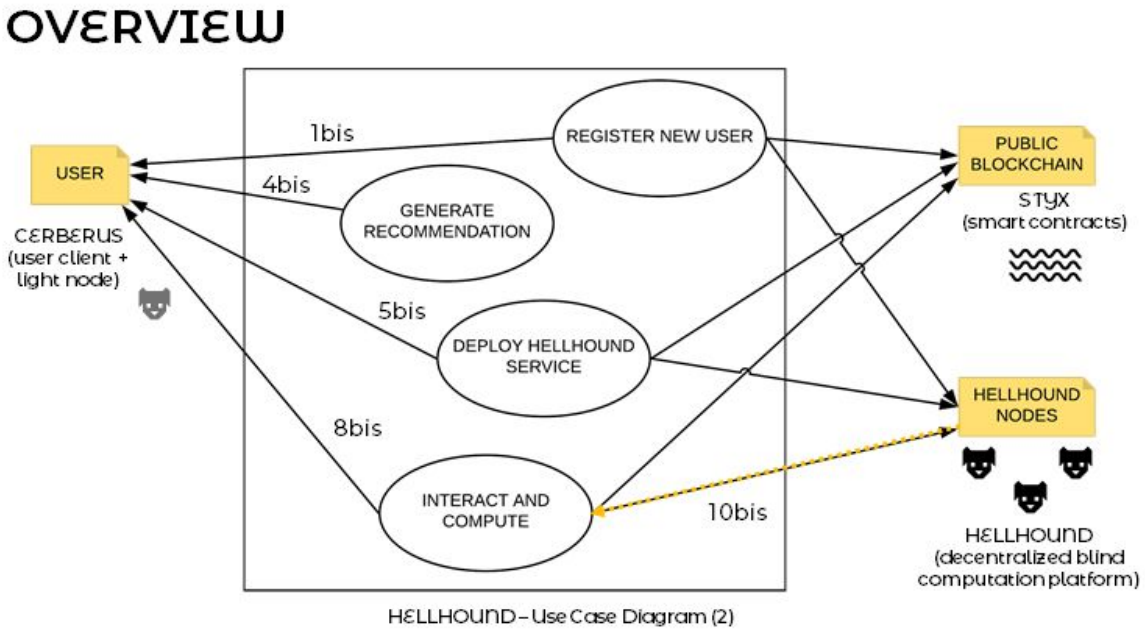
Use Case Diagrams



1. Interaction to create an authentication token for the user client (Cerberus) so that its TX are recognized by the HellHound network
2. STYX register is updated with the new user information
3. Each HellHound node updates its users register with token information for future secure interaction with Cerberus
4. User fills information needed for Pythia and sign it. This information is parsed to create a query and the recommendation engine generate a recommendation using its knowledge database
5. User confirms deployment of the cryptosystem recommendation and sends token to an escrow STYX contract
6. Metadata+QoS engagement+UserSIG are stored onchain for Service Level Agreement, other informations as well: selection of nodes + configuration specifications +

cryptosystem selected + all user inputs from step4. User tokens are stored in the escrow STYX contract.

7. VMs are built, Cryptosystems are set, job queue of HellHound nodes is updated.
8. Cerberus starts participating to the computation.
9. Cerberus interacts with STYX (onchain) to post proofs for verification
10. Cerberus interacts with HellHound nodes for multi party computation (if the cryptosystem is non-MPC, this step is skipped)
11. HellHound nodes interacts with STYX to post proofs for verification and thus be able to receive their reward once the computation is completed



1bis. Interaction, the user client (Cerberus) receives its authentication token
4bis. User receives the recommendation from Pythia and the associated Quality of Service (depending on the number of inputs and the requirements on privacy or time of computation and interaction [see more criterias used by Pythia in the CryptoMap paper]
5bis. Cerberus is informed of the status of the anonymized participating nodes (hash) - useful for UI component of Cerberus (dashboard)
8bis. Cerberus interacts with HellHound nodes for MPC
10bis. Hellhound nodes interact with Cerberus for MPC

Appendix B

Definition 1 : An homomorphic public-key encryption scheme on M (message space) is A quadruple (K, E, D, A) of probabilistic, expected polynomial time algorithms, satisfying the following:

- ***Key Generation:*** On input $1^{\wedge\sigma}$, the algorithm K outputs a key pair (k_e, k_d) where K denotes the key space and σ security parameter.
- ***Encryption:*** On inputs $1^{\wedge\sigma}, k_e$, and $m \in M$, the encryption algorithm E outputs a ciphertext $c \in C$, where C denotes the ciphertext space.
- ***Decryption:*** The decryption algorithm D is deterministic. $\forall\ m \in M$, it holds: if $c = E(1^{\wedge\sigma}, k_e, m)$, then $Prob\left[D(1^{\wedge\sigma}, k, c) \neq m\right]$ is negligible.
- ***Homomorphic Property:*** A is an algorithm that on inputs $1^{\wedge\sigma}, k_e$ and elements $c_1, c_2 \in C$, outputs an element $c_3 \in C$ so that $\forall m_1, m_2 \in M$:

  If $m_3 = m_1 \circ m_2$ and $c_1 = E(1^{\wedge\sigma}, k_e, m_1)$ and $c_2 = E(1^{\wedge\sigma}, k_e, m_2)$, then $Prob\left[D(A(1^{\wedge\sigma}, k_e, c_1, c_2) \neq m_3\right]$ is negligible.

What is Quantum computing?

A state in a classical computers is represented by bits which have either zero or one value. Quantum computers on the other hand use “qubits” where a single qubit can be encoded to multiple quantum superposition states. These unique properties of qubits allow algorithms to run significantly faster on a quantum computer than on a classical one.