GENETANK PLATFORM FLOWCHARTS

[Document subtitle]

CREATED: NOVEMBER 25, 2017

UPDATED: DECEMBER 4, 2017

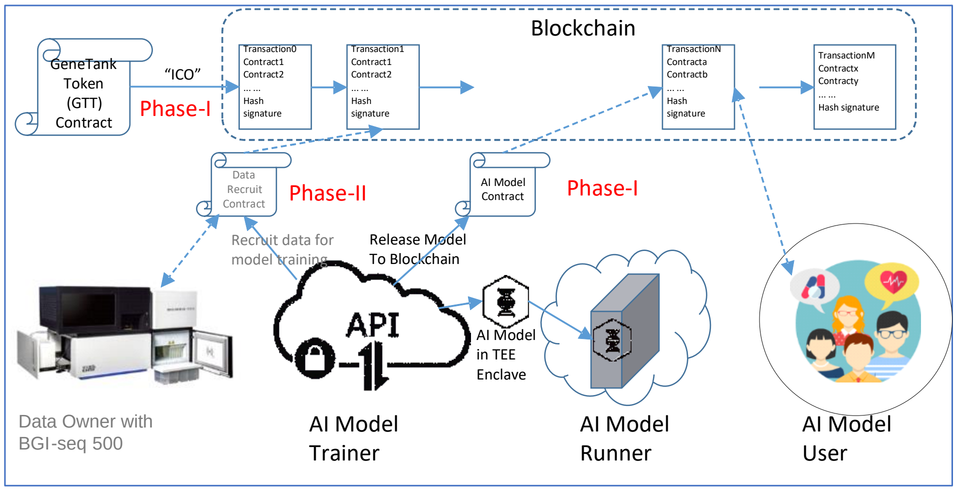
GENETANK INC
Cambridge, MA

Deleted: NOVEMBER 25, 2017

Genetank Platform Flowchart

1. Platform Overview

GeneTank GWAS AI Model Sharing Platform is a distributed blockchain based genetic and health data sharing and AI prediction service platform. The overview of the platform is shown in the below picture.



Blockchain
GeneTank Token (GTT) Contract
"ICO"
Phase-I
Transaction0 Contract1 Contract2 … … Hash signature
Transaction1 Contract1 Contract2 … … Hash signature
TransactionN Contracta Contractb … … Hash signature
TransactionM Contractx Contracty … … Hash signature
Data Recruit Contract
Phase-II
AI Model Contract
Phase-I
Recruit data for model training
Release Model To Blockchain
API
AI Model in TEE Enclave
Data Owner with BGI-seq 500
AI Model Trainer
AI Model Runner
AI Model User

Figure. Overview of Genetank Platform

The platform uses Ethereum blockchain as the backbone to integrate all the data and services into an anonymous, secure, privacy protected, and open system.

The blockchain is a distributed and virtual database. The information on the blockchain is open to all the participants and is protected by cryptography and continuously verified by participants. Therefore, the information on the blockchain can be trusted.

The platform consists of a Genetank Token, a Model sharing, and a genetic/health data sharing subsystem. We will develop the Genetank Token and Model sharing subsystem in the phase-I. The genetic/health data sharing will be developed in phase-II.

The Model sharing subsystem supports three operations:

1. New Enclave Version Releasing:

. Register the new enclave version to the Genetank Model sharing smart contract

Deleted: Model

Deleted: Create a contract for the model

. The auditors audit the new enclave version and send signatures of the auditing package to the Genetank Model sharing smart contract

Deleted: Create a dapp for model running and querying

2. Model Installing:

. The runner downloads the SGX enclave and installs on a computation platform to create an enclave instance.

Deleted: can

Deleted: their

. The runner registers the enclave instance to the Genetank model sharing smart contract

Deleted: on

. The trainer transfers the latest version of AI model to the enclave

Deleted: as a runner

3. Model querying:

. The user pays query fee deposit to the smart contract, and adds a hash of the genetic/health data for querying and her/his Ethereum address to the smart contract

Deleted: gets the address of runner,

Deleted: gets credential for SGX enclave from the trainer

. The user queries the model in the enclave instance

. The model trainer and runner get payed according to the contract if the query succeeds.

This document will focus in the detail flows of this operations.

2. Background Information

2.1 Development Framework

From developer point of view, the platform has some subsystems which need to be developed and some others which are used for the development and platform running.

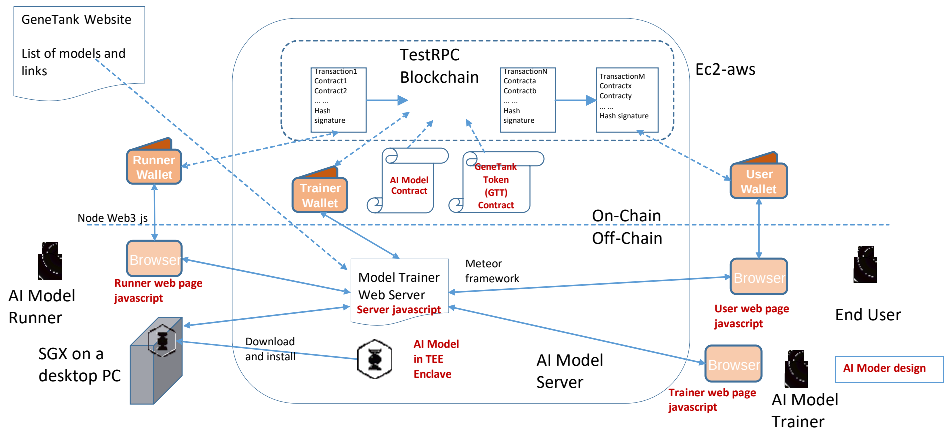


Figure. Development Framework

The subsystems to be developed include the AI Model, Genetank Token, Model sharing. The components that we are going to use are the TestRPC (private test Ethereum blockchain), wallet software (metamask --TBD), Genetank website, and browser (Chrome).

Deleted: need

The Model sharing subsystem includes a model trainer web server and the backend and frontend html, javascripts, and other data for the model trainer, model runner, and end user.

2.2 SGX Enclave Attestation

An SGX enclave is a Trusted Execution Environment (TEE) with Intel hardware enforcement. The purpose of SGX Enclave attestation is to verify the enclave is origin from the enclave developer without being tampered. The safety of the enclave design is not guaranteed by attestation. It can be ensured by auditing.

Deleted: developed by

Deleted: but it is out of the scope of this document

The flow of SGX Enclave attestation is shown in the below figure.

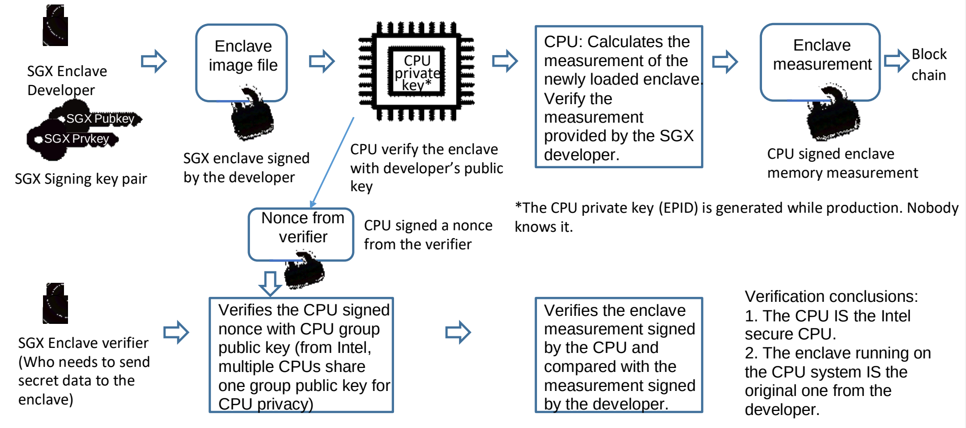

SGX Enclave Developer
SGX Pubkey
SGX Prvkey
SGX Signing key pair
Enclave image file
SGX enclave signed by the developer
CPU private key*
CPU verify the enclave with developer's public key
CPU: Calculates the measurement of the newly loaded enclave. Verify the measurement provided by the SGX developer.
Enclave measurement
Block chain
CPU signed enclave memory measurement
Nonce from verifier
CPU signed a nonce from the verifier
*The CPU private key (EPID) is generated while production. Nobody knows it.
SGX Enclave verifier (Who needs to send secret data to the enclave)
Verifies the CPU signed nonce with CPU group public key (from Intel, multiple CPUs share one group public key for CPU privacy)
Verifies the enclave measurement signed by the CPU and compared with the measurement signed by the developer.
Verification conclusions:
1. The CPU IS the Intel secure CPU.
2. The enclave running on the CPU system IS the original one from the developer.

Figure. The SGX Enclave Attestation Flow

The above part of the picture is the signing flow which includes the signing of the enclave image file and the signing of the running image of enclave. The signatures of the image file and the running image must match each other. The signature can be put on the blockchain and ready to be used for attestation.

Deleted: Both

Deleted: s

Deleted: must

The below part of the picture shows the attestation operations of the verifier who needs to render her/his secret information to the enclave. Firstly, the verifier certifies that the CPU which runs the enclave is an Intel secure CPU. Secondly, the verifier ensures the signature provided by the CPU matching the signature from the blockchain.

Deleted: provides the signature of

Deleted: running image

3. The Flowcharts for Model Sharing

The model sharing is based on a smart contract. The smart contract supports:

- Providing storage for:
  - Hashes of the auditing package and the attestation package of the Enclave versions
  - Signatures of auditors
  - Hashes of Enclave instances' public keys
  - Hashes of AI Model versions

Formatted

    - Querying deposits, etc.
- Providing codes for:
    - Registering enclave versions, enclave instances
    - Depositing querying fee for a query
    - Canceling a query
    - Closing a query (after the user confirms query is done or time-out)
    - Auditing the safety of the enclave

For phase-I, there are three main flows: new enclave version releasing, new enclave instance registering, new model version registering, and end user querying.

Deleted: an

Deleted: an

Deleted: Requesting a model access code (d

Deleted:)

Deleted: (phase-II)

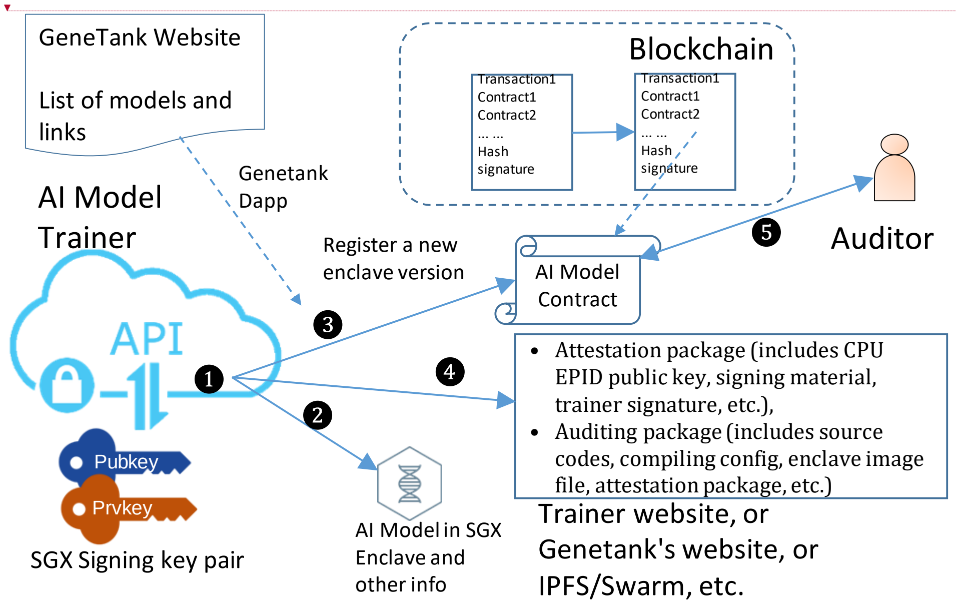
Deleted: model

Deleted: llation

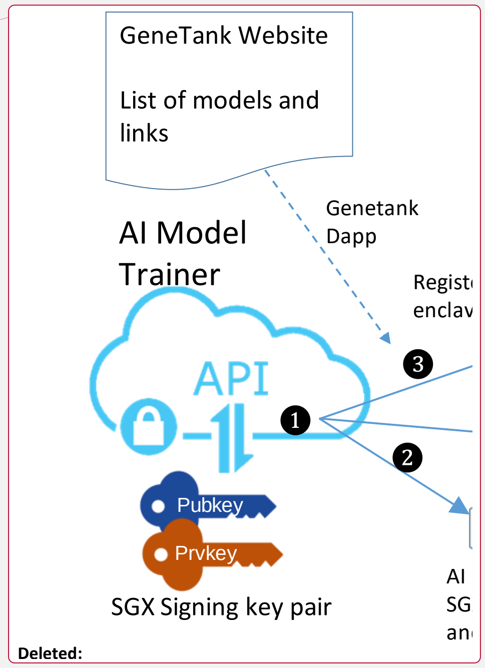
3.1 New Enclave Version Releasing Flow

One trainer Ethereum address can only register one enclave. One enclave can have multiple versions. One enclave can run one package of AI models. The AI models (parameters or coefficients) can have multiple versions.

The new model releasing flow is shown in below figure.



- Attestation package (includes CPU EPID public key, signing material, trainer signature, etc.),
- Auditing package (includes source codes, compiling config, enclave image file, attestation package, etc.)



Deleted:

Figure. The New Enclave Releasing Flow

The steps in the flow:

❶. The AI model trainer develops the enclave including the AI model codes in the enclave and trains the AI model

❷. The trainer hardcodes the her/his Ethereum address into the SGX enclave, generates an attestation package (includes CPU EPID public key, signing material, trainer signature, etc.), prepares an auditing package (includes source codes, compiling config, enclave image file, attestation package, etc.).

❸. The trainer uses Genetank AI Model sharing Dapp to register the enclave to the smart contract as a new enclave version. The registered info of the enclave includes a hash of the attestation package and the auditing package

❹. Upload the attestation and auditing packages onto the trainer's website and make all this information available to public (upload to trainer or Genetank's website, or upload to IPFS/Swarm, etc.)

❺. The auditors download the source codes and the compiling config, review the source codes, compile the codes, calculate the signing material, verify the trainer's signature, and sign the hash of the auditing package if the audit is passed. The signatures of auditors are added to the smart contract by the auditors.

The security operations in the model releasing flow:

Step #	Operator	Purpose	Security threats	Security Operations
2, 3	Trainer	Ensure the authenticity and origin of the enclave	The enclave could be tampered or replaced by malicious enclave.	Use the trainer's SGX signing key to sign the enclave and put the signature of the enclave onto the blockchain. So that everyone wants to use the enclave can verify the enclave.
				The trainer must keep the SGX signing private key in safe place.
7	3rd party	Enclave can keep user data safe in any operations	The enclave could have flaws or backdoors which can compromise customer data.	security 3rd parties audit the security of the enclave and add audit certificate to the smart contract. The audits includ . Review the source code . verify the source code and enclave image are identical

Deleted: re are 7

Deleted: encapsulate

Deleted: trainer

Deleted: a

Deleted: package.
G

Deleted:

Deleted: the signing material for the enclave. Sign the enclave with trainer's SGX signing key pair to generate enclave signature.

Deleted: the enclave's source codes, compiling configuration, signing material, and the signature of the trainer

Deleted: One trainer Ethereum address for one family of enclave + model software. One family of software means multiple versions of the software. One family of software can have multiple version model parameters can have multiple version.

Deleted: source codes, compile configuration, enclave image file, signing material, signature

Deleted: for auditors, runners, and users.

Deleted: leak

Deleted: by software flaws

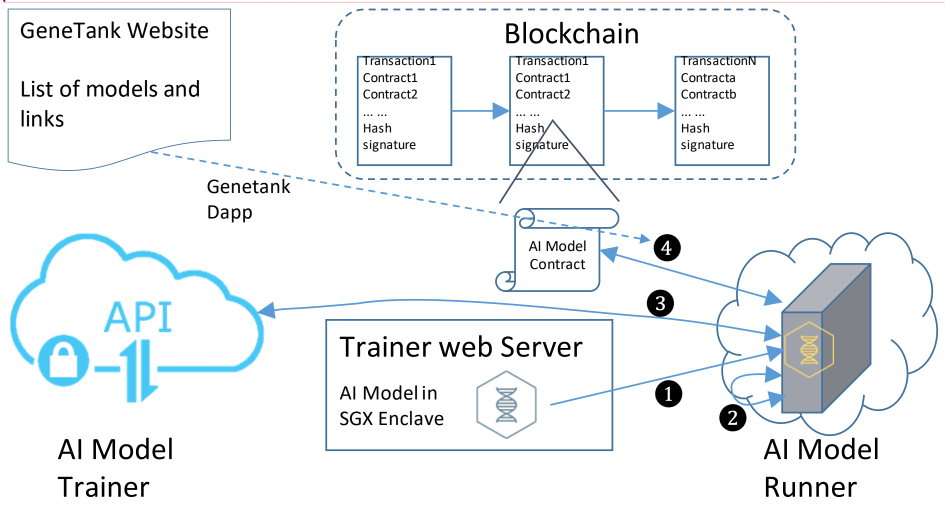
The signing material of the enclave includes the following fields:

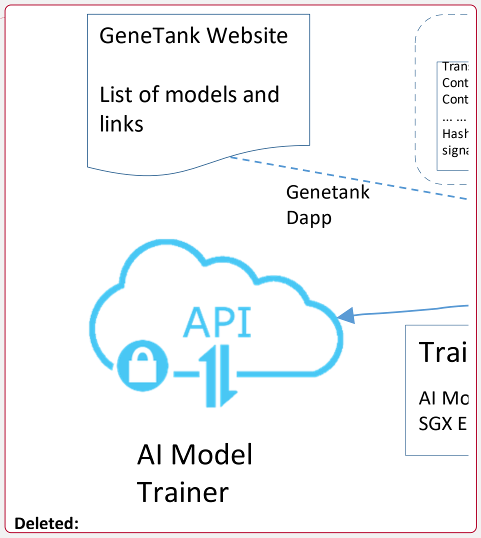
- Enclave Measurement (MRENCLAVE) – A single 256-bit hash (like a checksum) that identifies the code and initial data to be placed inside the enclave.
- The Enclave Author’s Public Key -- a hash of the enclave author’s public key (MRSIGNER).
- The Security Version Number of the Enclave (ISVSVN).
- The Product ID of the Enclave (ISVPRODID).

Deleted: similar to

3.2 New Enclave Instance Registering Flow

There are three steps in a new enclave installation registering flow as the figure below:





Deleted:

Figure. New Enclave Instance Registering Flow

The steps are:

❶. The model runner downloads the SGX enclave from the model trainer’s web page

The model runner verifies the info of the enclave with the hash obtained from the blockchain.

❷. The model runner runs the enclave to create a new enclave instance and assign the runner’s account address to the enclave.

The model runner installs AI models to the enclave.

❸. The runner informs (TBD: broadcast or point to point? How to do point to point communication?) the model trainer to do attestation on the new instance. The model trainer attests the new enclave instance. If the attestation succeeds, the trainer transfers the latest AI model version to the enclave.

Deleted: provides an enclave public key certificate signed by the trainer’s Ethereum account’s private key to the runner.

❹. The runner registers the new enclave instance to the model smart contract.

Deleted: uses the certificate to

Deleted: The trainer’s wallet address, the runner’s wallet address, the enclave’s public key, enclave version, and model version must be added to the smart contract.

Deleted: ❺ The auditors audit the new enclave, and register the audit result to the smart contract.

The security operations in the flow:

Step #	Operator	Purpose	Security threats	Security Operations
1	Runner	Safety of the smart contract	Fake fishing smart contract	Check the security audit certificate in the smart contract.
1	Runner	Download the enclave image.	Malicious websites provide fake enclave image.	Use the enclave signature on the blockchain to load the enclave. If the signature doesn’t match, the loading of the enclave will fail.
2	Runner, Trainer	The runner runs the enclave. The trainer attests the running enclave.	The runner could be the malicious who runs a fake enclave.	The trainer (EPID verifier) verify the runner (EPID member)’s EPID with the help of Intel facility (EPID issuer) *
3	Runner, Trainer	Register the runner to the smart contract	Fake runner could register to the smart contract	The trainer provides a certificate to runner for the runner’s registration
* Possible trust policies: only trusting a specific version of an enclave, identified by the measurement of the code and data in the enclave, or trusting all enclaves with a specific Product ID from a specific enclave author, identified by the hash of the public key in the ISV certificate.				

3.3 Model Querying Flow

When an end user wants to query an AI model for disease prediction, the following flow will be used.

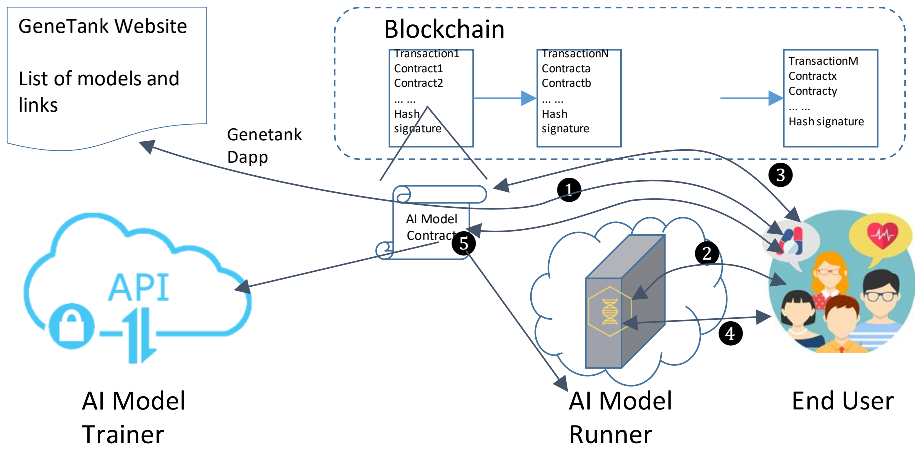


Figure. The Model Querying Flow

The steps in the flow are:

❶. The users access the Genetank webpage to get the dapp and access the AI model smart contract.

❷. The user attests an SGX enclave instance at least once before sending any private info to the it. If the attestation succeeded, the user can store the public key of the enclave and use it for future authenticate the enclave.

❸. The user adds some GTTs as secure deposit for using an pre-determined enclave instance and a hash of querying data to the smart contract. The combination of the deposit ID and the query data hash is used as an access code for user to access the AI model in the enclave.

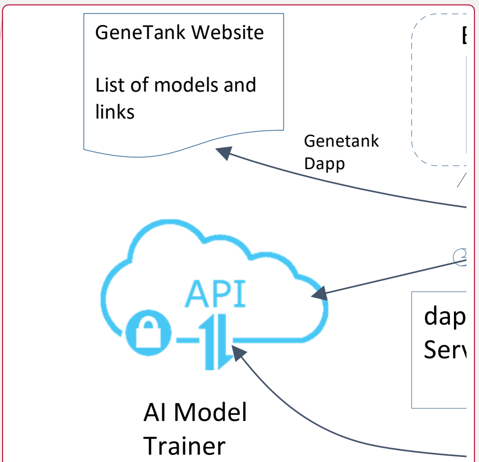
❹. The user calls the AI model with the access code as parameter to get prediction.

The untrusted part of SGX application can access the blockchain and provides all the blocks after the model sharing smart contract to the enclave for verification. When a user provides a querying request with an access code, the enclave will verify the block contain the transaction of paying secure deposit. If the deposit is valid, the enclave ID is matched, and the hash of the querying data is matched, then the enclave will execute the AI model. Otherwise it will refuse the request.

The access code is not possible to be double used because it can only be used on one enclave instance for a query data. The enclave can stored the access codes which has been used.

❺ The user confirms the transaction or the trainer confirms the transaction when timeout, the secure deposit is distributed to the model trainer and runner.

If the user wants to cancel the querying, she/he must do it before timeout. The user must send a canceling request to the enclave. The enclave signs on the access code to indicate the access code hasn't been used. The user sends the enclave signed access code to the smart contract to cancel the query and gets the deposit back.



Deleted:

Deleted: i

Deleted: verifies the enclave public key from the blockchain and authenticates

Deleted: the

Deleted: it creates an encrypted link between the end user and the enclave.

Deleted: If the user doesn't have enough GTT, he can buy GTT from Genetank.

Deleted: The user uses the deposit ID to apply an access code from the trainer. The trainer signs the access code and sends it to the user.
❺ The user sends the access code to the runner. The runner verifies the access code with the deposit information on the blockchain, signs the access code, and sends it back to the user.
❻

Deleted: enclave verifies the signatures of the trainer and the runner on the access code. (TBD: Should the access code be signed by Genetank to have central control on all querying transactions? If the access code has to be signed by Genetank, then the trainer and runner signatures are not necessary because they can trust Genetank.)

Deleted: The enclave checks the one-time-usage code in the access code to ensure one time usage. The query result will be returned to the user directly.
❼

Deleted: must

Deleted: trainer

Deleted: do the cancellation

The security operations in the flow:

Step #	Operator	Purpose	Security threats	Security Operations
1	User	Access to a dapp	The html/js of the dapp could be fake.	Use https to access to the trusted website to access the dapp. The certification chain of the website must include the model trainer.
1	User	Access to a dapp	The enclave could be fake (for data fishing)	The smart contract must be certified by 3rd party and the enclave is safe.
3	Runner, Trainer	Register the runner to the smart contract	Fake enclave could be registered to the smart contract	The trainer provides a certificate for attested enclave to runner for the registration

3.4 Model Version Upgrading Flow

1. When a new model version has been trained by a trainer, the trainer need to upload the model onto the trainer website, make it available for all the model runners.
2. The trainer registers the new model version on the smart contract
3. The runners get notice of new model version from the smart contract (TBD: through event?) and decide when to upgrade their models
4. The runners download the models, verify the hash of the new model version on the blockchain, send the model to the enclave for updating, update the model version info of the enclave instance on the blockchain.

The above the runner steps could be automated with Genetank Model sharing Dapp.