GENETANK PLATFORM FLOWCHARTS

[Document subtitle]

NOVEMBER 25, 2017

GENETANK INC
Cambridge, MA

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.

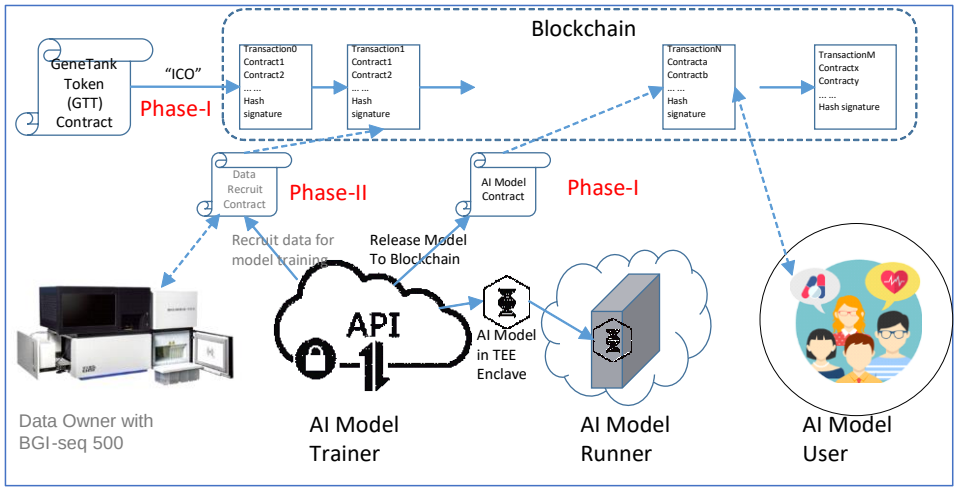


Figure. Overview of Genetank Platform

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. Model Releasing:

. Create a contract for the model

. Create a dapp for model running and querying

2. Model Installing:

. The runner can download the SGX enclave and install on their computation platform

. The runner registers on the contract as a runner

3. Model querying:

. The user gets the address of runner, pays query fee to the contract, and gets credential for SGX enclave from the trainer

. The user queries the model

. 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.

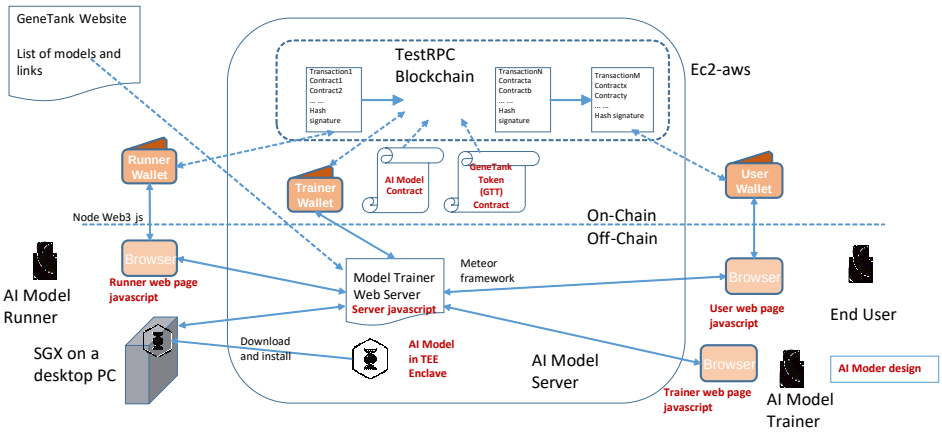


Figure. Development Framework

The subsystems need 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).

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

SGX is a Trusted Execution Environment (TEE) developed by Intel. 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 audit but it is out of the scope of this document.

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

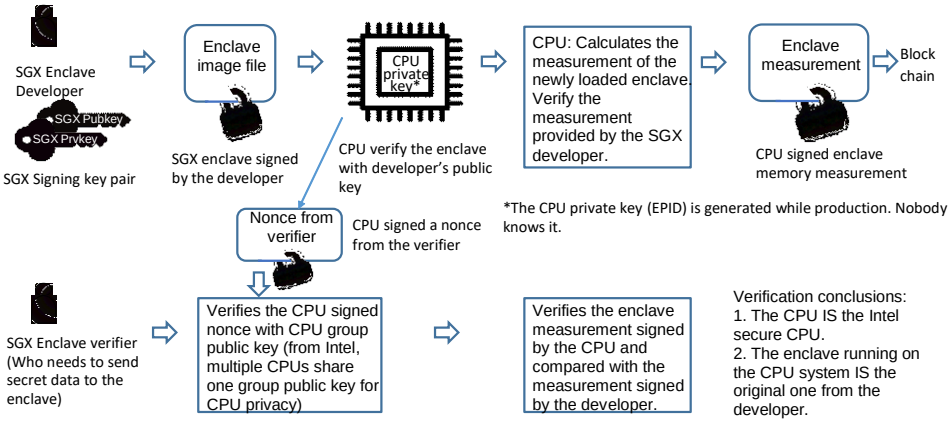


Figure. The SGX Enclave Attestation Flow

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. Both signatures must be put on the blockchain and ready to be used for attestation.

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 provides the signature of the enclave running image is an Intel secure CPU. Secondly, the verifier ensures the signature provided by the CPU matching the signature from the blockchain.

3. The Flowcharts for Model Sharing

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

- Providing storage for:
- Providing codes for:
  - Registering an enclave version, an enclave instance
  - Requesting a model access code (depositing querying fee) for a query
  - Canceling a query

Deleted: <#>A signature of the enclave image file from the model developer
<#>A list of registered runners
<#>A signature of source codes and building script of the enclave (phase-II)
<#>A list of auditors (phase-II).

Deleted: runner

- Closing a query (after the user confirms query is done or time-out)
- Auditing the safety of the enclave (phase-II).

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

3.1 New Enclave Version Releasing Flow

The new model releasing flow is shown in below figure.

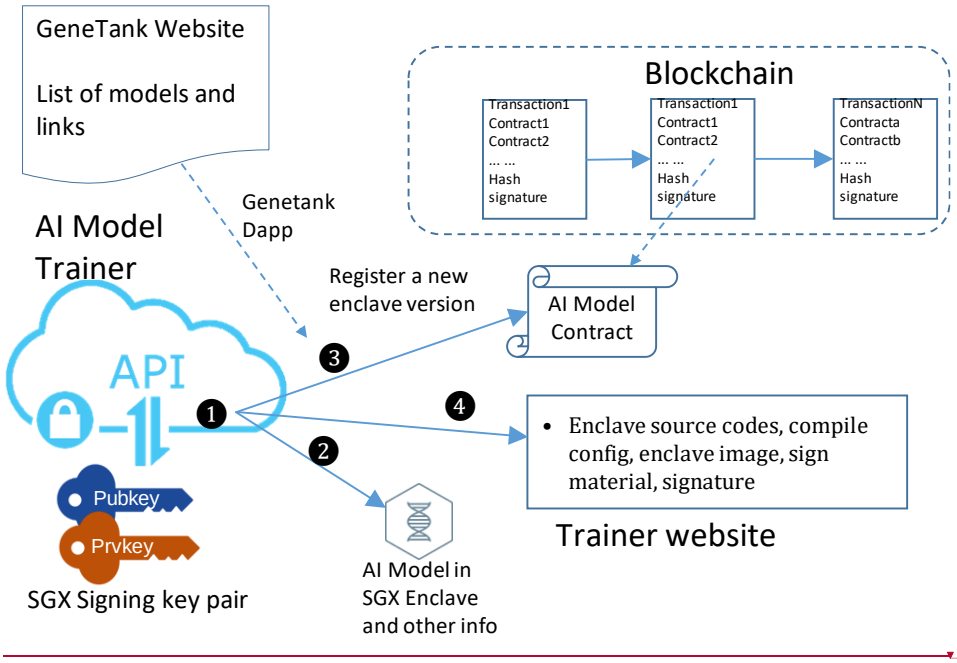


Figure. The New E Releasing Flow

There are 7 steps in the flow:

❶ The AI model trainer trains the model

❷ The trainer encapsulate the trainer Ethereum address into a SGX enclave package.

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

Deleted: Model

Deleted: Model

Deleted: E

Deleted: trained model and

Deleted: /id

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 include hash of the enclave's source codes, compiling configuration, signing material, and the signature of the trainer.

❹ Upload the source codes, compile configuration, enclave image file, signing material, signature onto the trainer's website and make all this information available for auditors, runners, and users.

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 leak customer data by software flaws.	security 3rd parties audit the security of the enclave and add audit certificate the smart contract. The audits include: . Review the source code . verify the source code and enclave image are identical

The signing material of the enclave includes the following fields:

- Enclave Measurement (MRENCLAVE) – A single 256-bit hash (similar to 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: Use genetank AI model smart contract template to create a smart contract (supports runner registration, model querying, and payment sharing).

Deleted: I

Deleted: in the smart contract

Deleted: Develop html/javascript for runners and users to access the smart contract and SGX enclave

Deleted: ❺ Run the dapp .
❻ Migrate the contract to the Blockchain .
❼ Open the source code of the enclave to a public website. 3rd party audit and certificate. (Phase-II) .

Deleted: accident

Deleted: signature

3.2 New Enclave Instance Registering Flow

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

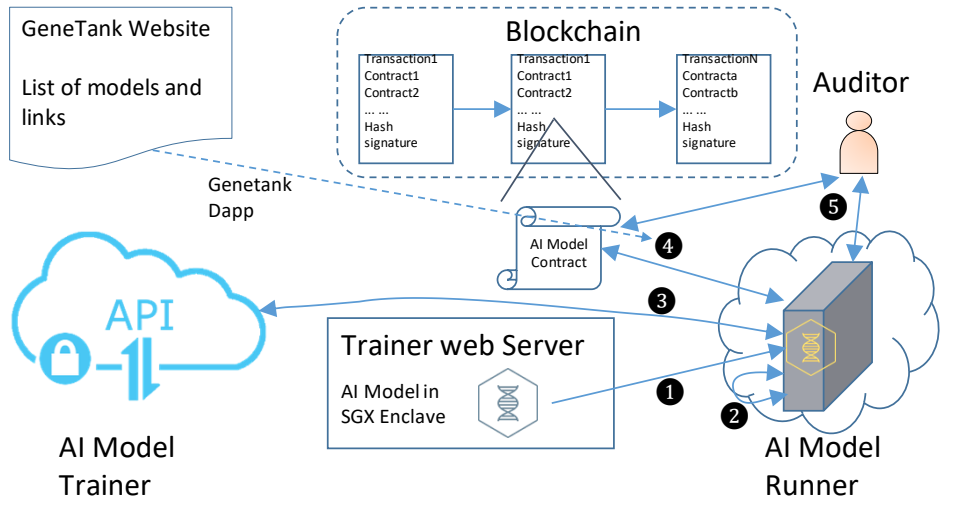


Figure. New Enclave Instance Registering Flow

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

❹ The runner uses the certificate to register the new enclave instance to the model smart contract. 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.

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

Deleted: Installation

Deleted: Installation

Deleted: dapp

Deleted: calculates the checksum of the enclave and verify with the checksum on the smart contract

Deleted: R

Deleted: provides information to the model trainer

Deleted: authenticates the security of the

Deleted: it is safe,

Deleted: the

Deleted: runner

Deleted: ❸

Deleted: himself

Deleted: and

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 provide 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.

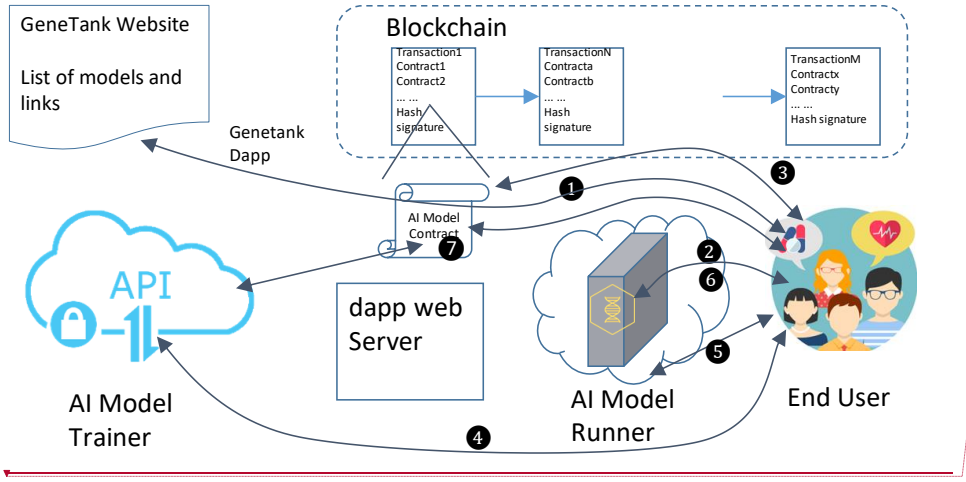


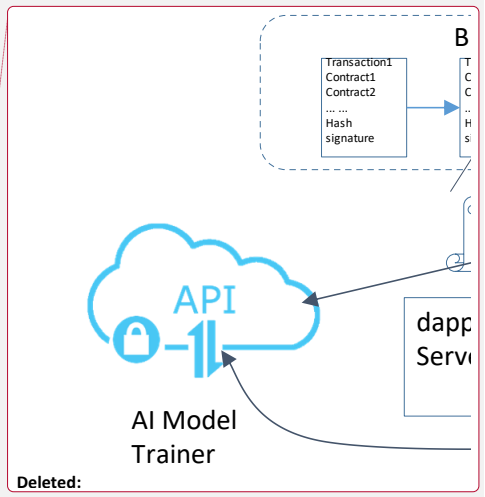
Figure. The Model Querying Flow

The steps in the flow are:

1. The users access the Genetank webpage to geit the dapp and access the AI model smart contract.
2. The user verifies the enclave public key from the blockchain and authenticates the SGX enclave. If succeeded, it creates an encrypted link between the end user and the enclave.
3. The user adds GTT as secure deposit to the smart contract. If the user doesn't have enough GTT, he can buy GTT from Genetank.
4. 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.
5. 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.
6. The user calls the AI model with the access code as parameter to get prediction.

   The 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.)

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



Deleted:

Deleted: html/javascript of

Deleted: use the js to

Deleted: use

Deleted: signature on

Deleted: to

Deleted: it is safe

Deleted: to

Deleted: accesses to the model smart contract to

Deleted: gets query

Deleted: is

Deleted: 6

If the user wants to cancel the querying, she/he must do it before timeout. The enclave must sign on the access code to indicate the access code hasn't been used. The user sends the enclave signed access code to the trainer to do the cancellation.

The security operations in the flow:

Deleted: If the user cancels the transaction, he needs send message to the trainer. The trainer revokes the access code then sends the deposit back to the user.

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

Deleted: runner

Deleted: runner's

3.4 Model 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 on the blockchain.

Formatted: List Paragraph, Numbered + Level: 1 + Numbering Style: 1, 2, 3, … + Start at: 1 + Alignment: Left + Aligned at: 0 cm + Indent at: 0.63 cm

The above the runner steps could be automated with Dapp.