

Genetank platform flowcharts

[Document subtitle]

November 25, 2017

Updated: December 4, 2017

Genetank Inc

Cambridge, MA

Genetank Platform Flowchart

# 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.

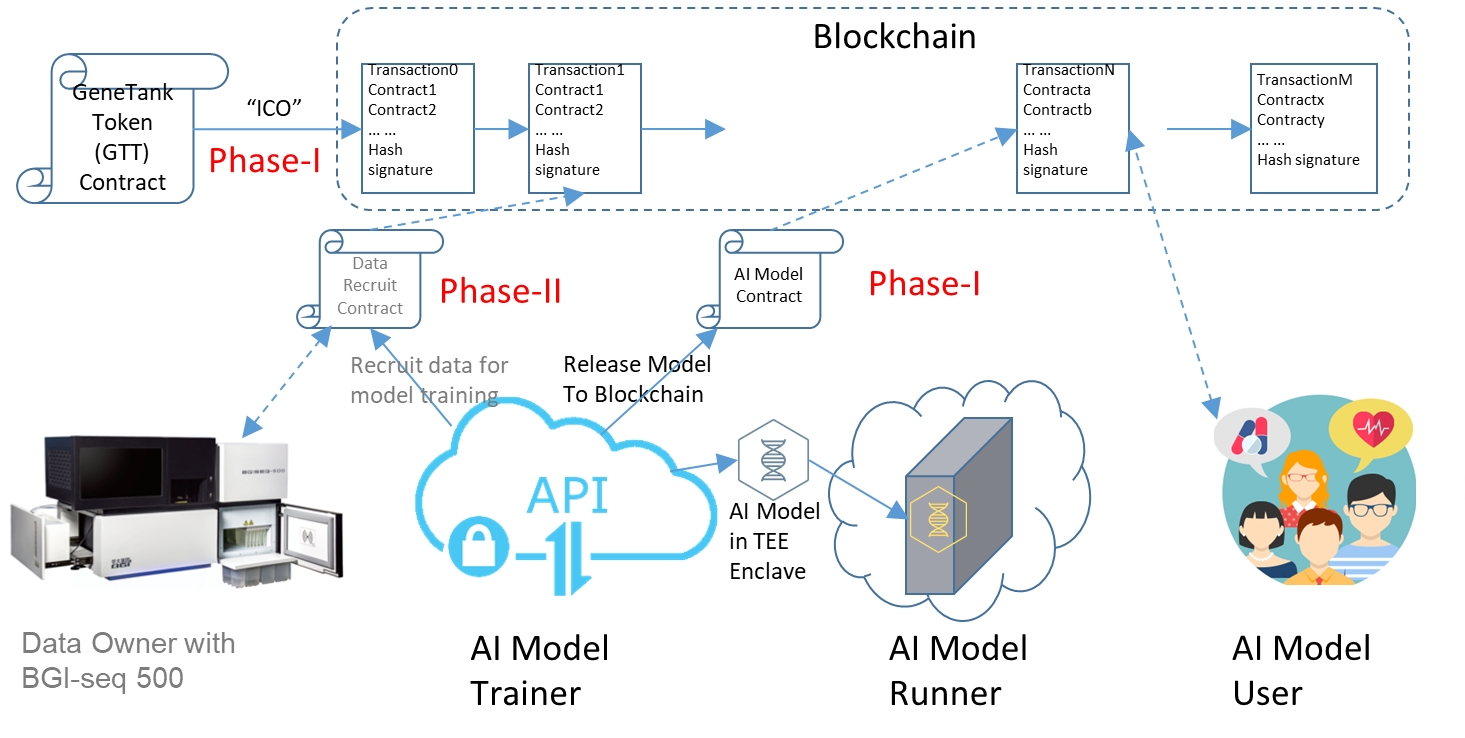


Figure. Overview of Genetank Platform

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. New Enclave Version Releasing:

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

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

1. Model Installing:

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

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

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

1. 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

. 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.

# 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.

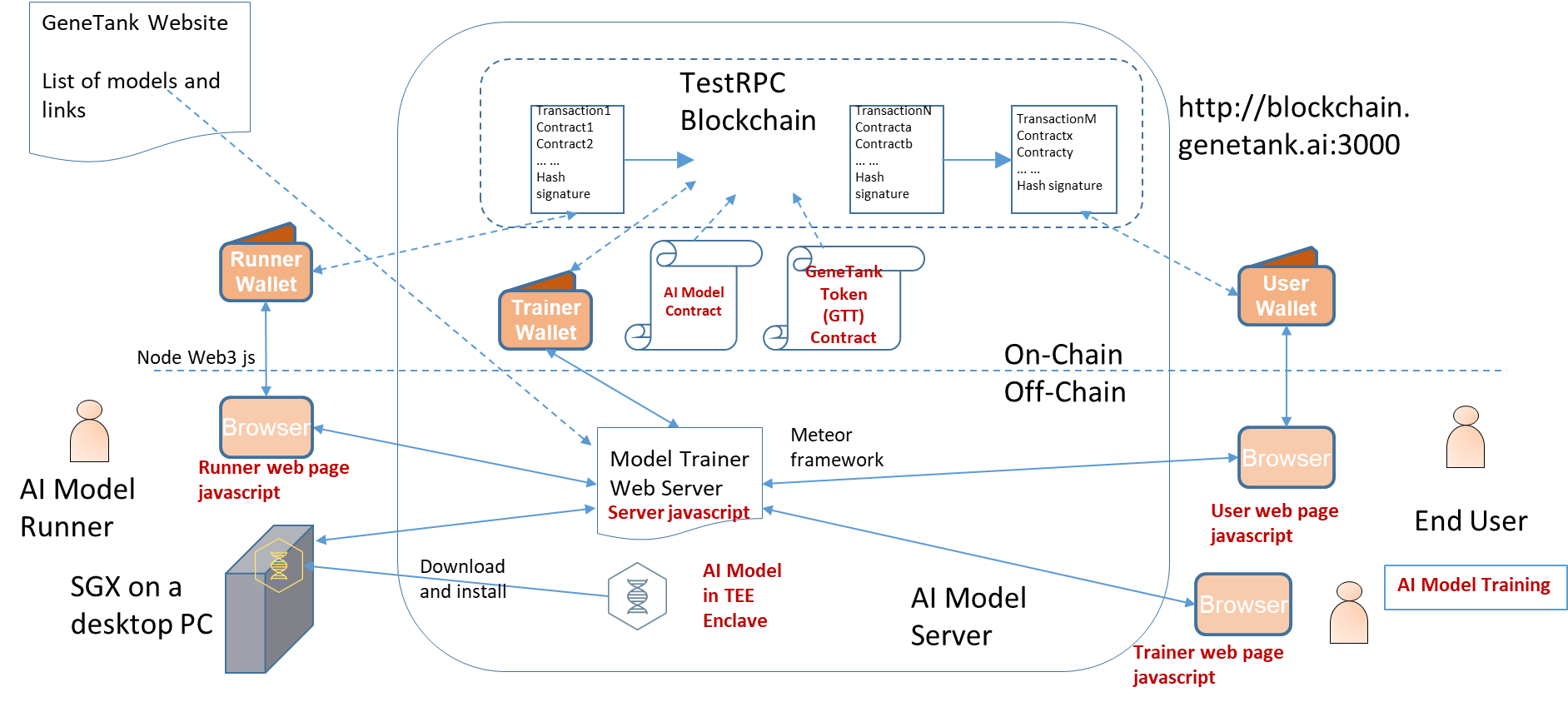


Figure. Development Framework

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).

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.

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

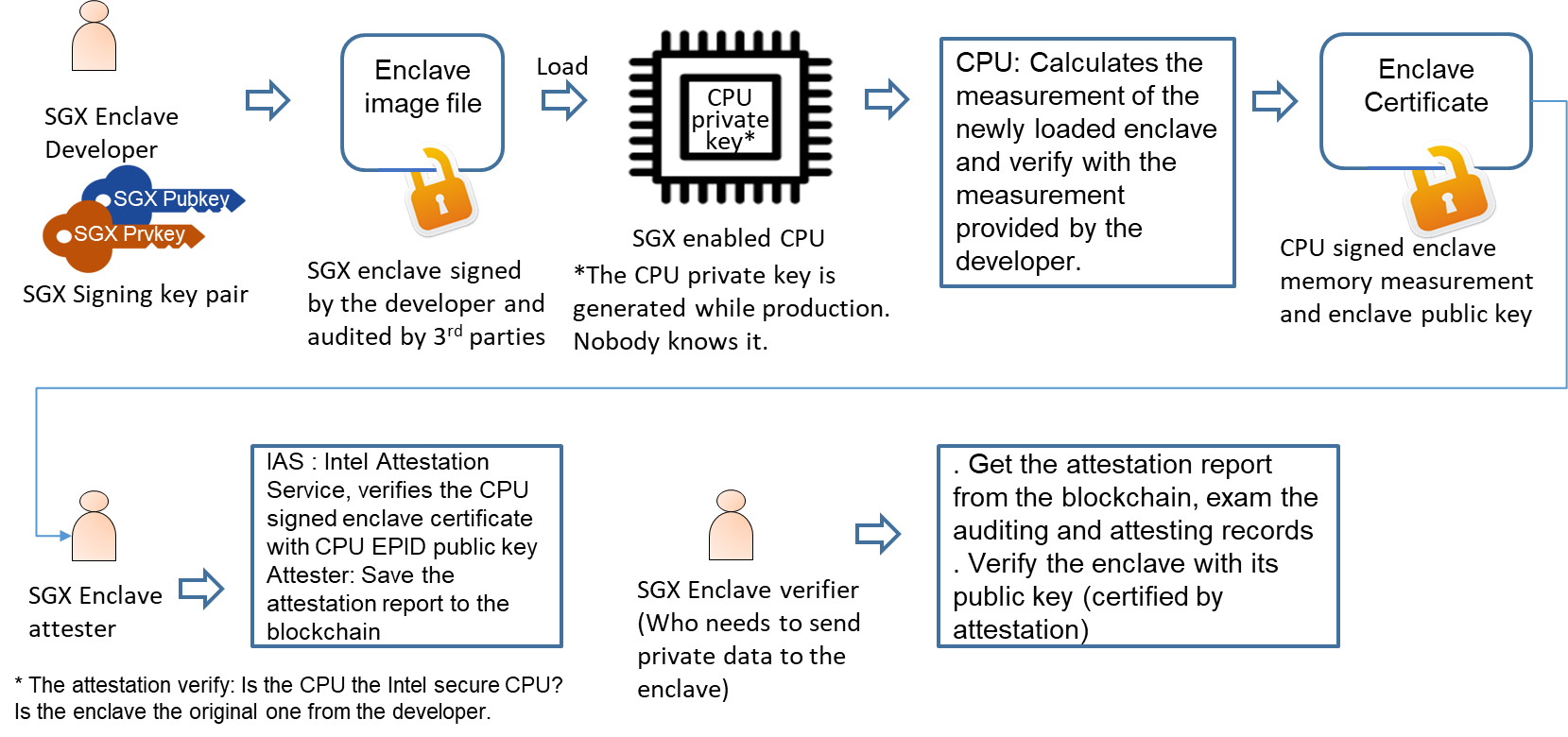


Figure. The SGX Enclave Attestation Flow

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.

The below part of the picture shows the attestation operations with the help of Intel Attestation Service (IAS). The attestation report can be put onto the blockchain. The verifier who needs to render her/his secret information to the enclave. Firstly, the verifier verifies the auditing and attesting records to make sure the public key of the enclave is valid. Secondly, the verifier verify the enclave with its public key.

# 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
  + 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.

## 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.

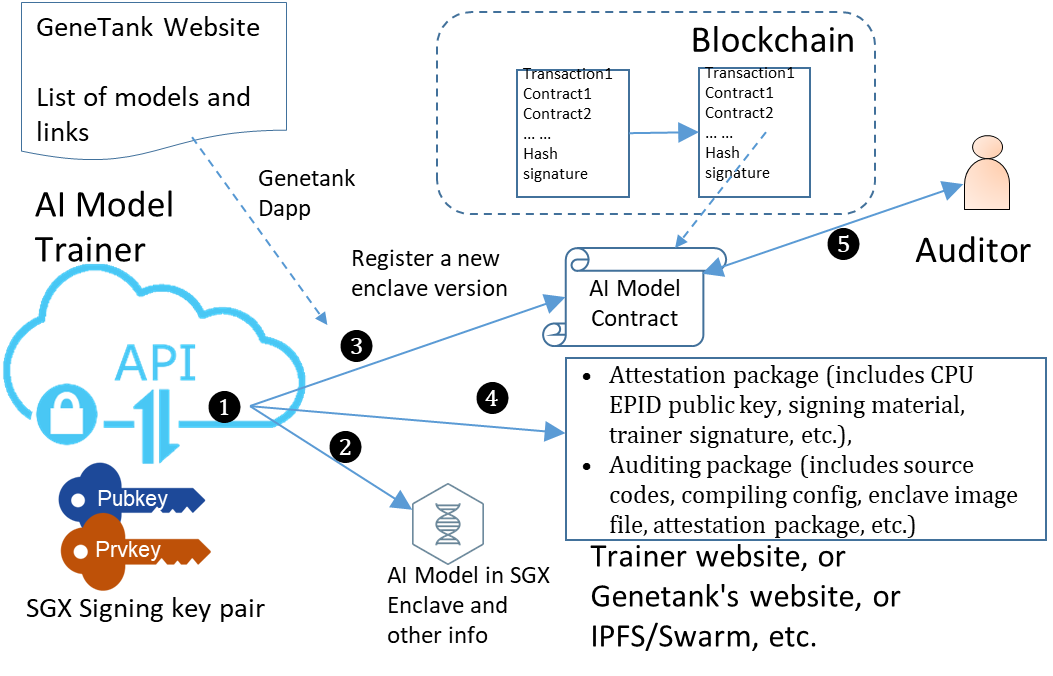


Figure. The New Enclave Releasing Flow

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 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 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 (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).

## 3.2 New Enclave Instance Registering Flow

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

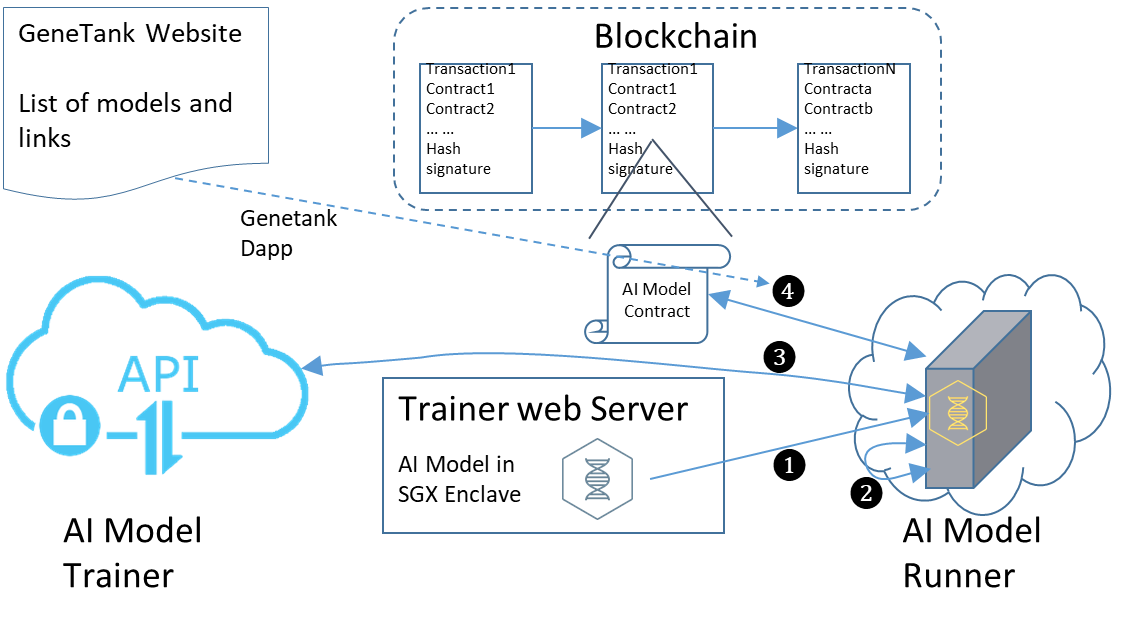


Figure. New Enclave Instance Registering Flow

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.

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 and sends the attestation report signed by Intel to the runner. The report includes the public key of the enclave, can be a certificate of the enclave provided by Intel.

❹. The runner verifies the attestation report received from the trainer and registers the new enclave instance to the model smart contract with the attestation report.

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.

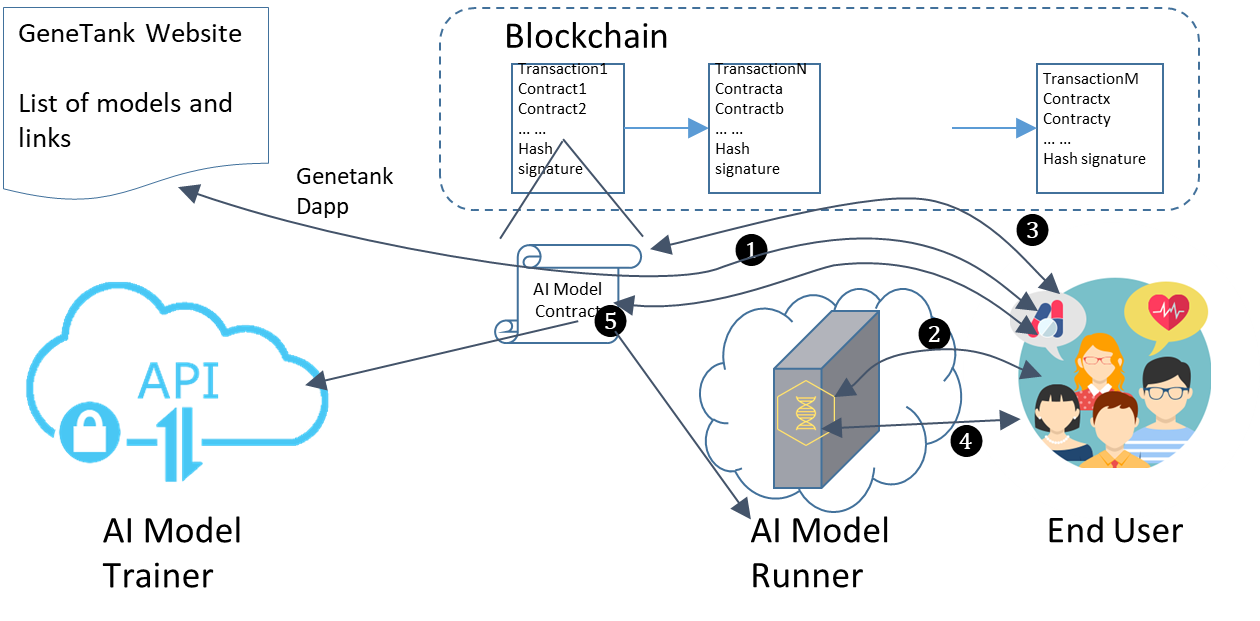


Figure. The Model Querying Flow

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 authenticates the SGX enclave the public key in the attestation report on the blockchain. If succeeded, it creates an encrypted link between the end user and the enclave.

The enclave generates an access code and sends it to the user.

❸. The user adds some GTTs as secure deposit for access code from then enclave. The user adds a hash of querying data to the deposit to prevent other people to use the access code. The combination of the deposit ID and the query data hash is used as an access code for user when querying 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 blockchains and provides all the blocks (at least from the model sharing smart contract to the most recent block) 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.

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.