# **Project** **Soteria Attack Scenario**

## Client-Side Components

* User
  + Opens the Query Service Client and requests the query service
  + Sends its query to the query service client
* Query service client
  + Authenticates the requested service
  + Extracts query predicates and builds the query
  + Encrypts the query predicate and sends it to the Service Query portal
* Encrypted Result store
  + Encrypted result is decrypted in the Trusted Execution Environment
  + Result is received by the query service client which sends result to the corresponding query user

## Server-Side Components

* Service Query Portal
  + Authenticates user and institution using
    - User at Institution
    - Query Id
    - Query Data
    - Job Id
  + Transmits user query to query engine
* Service registry
  + Validates institutions and possibly users
  + Validates that the Institution/user is authorized to ask the requested service.
  + On successful authorization forwards the users metadata
* Service KMS
  + Stores Institution id with Public and Evaluation keys (May be encrypted with HSM using symmetric key encryption like AES)
  + Provides (institution specific) Evaluation Keys
  + Encoding techniques is not known to the KMS
* Data Store
  + Stores database in cleartext or plaintext
  + Query predicates are searched in the database for results
* FHE Query Engine
  + Does homomorphic computation using:
    - User metadata
    - Institution eval keys
    - Data store
* Encrypted Results Store
  + Receives encrypted results from query engine
  + Allows retrieval of results by relevant institution

## Passive Risks: what is leaked when every party follows the protocol?

Blue highlights indicate risks when component can observe client behaviour. (CCA)

Green highlights indicate risks associated with corrupted ciphertexts, and the possibility of ciphertext verification attacks. (CVA)

* Service query portal
  + Learns frequency of queries by institution
  + Knows which institutions are valid
  + Knows which users are valid at a given institution
  + Processes a query
    - Knows the query predicates
    - Sees ciphertext of query predicate
* Service registry
  + Learns frequency of queries by institution and by user
  + Learns user/institution meta data
    - Learns institution security level
    - Learns dimension of user input?
    - Type of service user/institution is registered for?
* Service KMS
  + Learns frequency of queries by institution
  + Sees Eval keys: has access to a large quantity of ciphertexts
  + Keys can be encrypted using the HSM can only see the encrypted Eval keys
* Data Store
  + If can observe client behaviour, may be able to infer (something about) user’s data: however unclear whether this component knows which request corresponds to which user.
  + If plaintext (not cleartext) learns something about parameters of schemes used
* FHE Query Engine (closest to monolithic HE server from current modelling)
  + Learns frequency of queries by institution
  + Receives a query: same risks as service query portal
    - Learns the query predicates
    - Sees ciphertext of query predicates
  + Learns user/institution meta data: same risks as service registry
    - Learns service security level
    - Size of evaluation key / query predicates size
    - Learns dimension of user input.
    - Type of service user/institution is registered for?
  + Receives data: same risks as data store
  + Receives eval keys: same risks as service KMS
  + Sees result ciphertext
  + additional risks come from information seen in conjunction
    - Has query ciphertext AND eval keys
    - Sees database AND knows specific user
* Encrypted results store
  + Understands frequency of queries by institution and by user from retrieval
  + Sees result ciphertext
    - Can infer something about query evaluation circuit
    - Can infer about ciphertext decryption from user/institution behaviour

## Active Risks: what can be leaked when components deviate from the protocol description?

Blue highlights indicate risks when component can observe client behaviour.

Green highlights indicate risks associated with corrupted ciphertexts, and the possibility of ciphertext verification attacks.

* Service Query Portal
  + Denial of service attacks
  + Can substitute or modify query data ciphertext, possibly leading to decryption oracle creation and/or corrupted ciphertexts
  + Can change query logic
  + Can swap institution ID, leading to corrupted ciphertexts
* Service registry
  + Can change metadata, which can lead to corrupted ciphertexts
* Service KMS
  + Denial of service
  + Corruption of eval keys, leading to corrupted ciphertexts
  + Corruption of public encryption key, leading to corrupted ciphertexts
  + Can generate valid ciphertexts using public keys
* Data Store
  + Denial of service
  + Can modify database to try and infer something about query data by observing client behaviour
  + Can modify database to ensure decryption failures, leading to corrupted ciphertexts
* FHE Query Engine
  + All risks previously covered, and
  + Can reply chosen query request by valid ciphertext
  + Can generate any request and encrypt it
  + Can corrupt the ciphertext for the correct query
  + Can evaluate a different circuit other than the one requested
  + Can generate valid ciphertexts associated to the query ciphertext
  + Can produce corrupt result
* Encrypted Results Store
  + Can use ciphertext received metadata/context to generate a corrupted ciphertext
  + Can generate valid ciphertexts with known relationship to target by e.g. adding plaintexts and observe client behaviour to create a decryption oracle

**Categorization of attacks**

The attacks mainly can be categorized into

1. Passive attacks

* Here all the components follow the protocol however, they passively keep track of all steps followed and any private information that can be gained from it.
* To avoid such attacks the system must be designed that leaks as minimum information as possible.

1. Active attacks

* In this attack setting the components can deviate from the protocol and behave independently resulting into even wrong computation.
* These types of attacks are usually hard to take care of.

1. Side Channel Attacks

* These are the attacks that are based on the side information that are gathered when the protocol or the algorithm is implemented and used.
* Timing information, power consumption electromagnetic leaks and sound analysis are the examples of side information’s that are exploited to facilitate the side-channel attacks.

1. Denial of Service attack

* These are the attacks meant to shut down a machine or network, making it inaccessible to its intended users.
* DoS attacks are usually accomplished by flooding the target with traffic or sending it information that triggers a crash.
* In both instances, the DoS attack deprives legitimate users of the service or resource they expected.

1. Insider attack

* These are the attackers who have access to the inside leakage and can mount attack based on the information they have.

1. Combination of attacks

* In this attack techniques different attack gets combined and uses the combined information to extract the unauthorized information.

**Data Flow Diagram of Query Execution**

Diagram

Description automatically generated

Fig: Basic Data flow diagram of FHE query computation

Diagram

Description automatically generatedFig: Basic Data flow diagram of FHE query computation. Key and Metadata are stored along with its digital signature so that the KMS cannot fool users by sending wrong key or wrong Metadata.

Diagram

Description automatically generated

Fig: Query is signed by the query user to avoid query and query id malleability attack

Diagram

Description automatically generated

Fig: In Service Registry service metadata is stored in either of two way-- computing digital signature (institutional id, user id, service id, service name, HE para and HE clients) or encrypted all with attestation key of TEE to restrict service registry sending wrong metadata for FHE query computation.

**Diagram

Description automatically generated**

Fig: Service KMS are stored along with its digital signature computed as DS(Institutional Id, User id, Service Id, Service Name, HE Para, HE Client) to avoid Key malleability and wrong key received for computation.

Diagram, schematic

Description automatically generated

Fig: Database for computation is stored in the encrypted form encrypted with TEE attestation key to avoid Decryption failure due to wrong data provided for computation.

Diagram, schematic

Description automatically generated

Fig: To avoid query result malleability attack query result along with its digital signature is stored in the encrypted database which is later decrypted in the TEE.

**Possible Attacks in each Component**

**Client Service Portal**

* Denial of service attack
  + The user cannot perform any query on the database
  + The reason about the same is mentioned in the feedback reply send to the user with standard error code
* Genuine user authentication fails
  + The user cannot perform any query on the database
  + However, the user knows why the authentication is failed through the feedback received
  + The feedback message mentions the standard error code
* Requested service not provided to the user
  + The user cannot perform the intended service computation on the database
  + The reason is mentioned with standard error code in the feedback reply thus the user knows the reason behind the requested service fail.
* Client Service Portal can send (encryption key, metadata) in the following four possible ways
* Correct, Correct
* This is correct and creates no problem for query computation
* Correct, Wrong
* Wrong here means metadata is of another user
  + - If metadata is completely wrong, then it gets detected in the service query portal when user is verified
* The user encrypts his query and sends it for query computation
* Service query portal authenticates the user and sends it to FHE Query Engine
* Service registry authenticates the user if it has access for the service requested then he replies with the user metadata
* The service key is asked from service KMS using user’s metadata and service KMS replies with necessary keys
* FHE query engine performs query computation and stores it in the encrypted form
* Encrypted result is decrypted in the TEE and later it is sent to the user.
* However, the result obtained by the user is wrong due to wrong key used for query encryption in the first place itself.
* I think this form of attack can be restricted by storing metadata and key along with its DS.
* Wrong, Correct
* Using the same techniques as that of above this form of attack can be identified and restricted
* Wrong, Wrong
* He gets detected in the service query portal when the user gets verified
* Malleability of query attack
  + Query malleability can be restricted using digital signature computed and send by the user along with the encrypted query
  + Service query portal verifies the received digital signature and if it finds any discrepancy with that of the received digital signature then it sends feedback to the user saying signature mismatch.
* Authorizes the unauthorized user and sends the correct metadata and key to him (Active)
  + This situation may arise when the query service client is completely compromised
  + Attacker who has key and metadata of a genuine user can’t be detected
  + He can perform the query computation without any problem and finally gets the result as a normal genuine user

**Service Query Portal**

* Denial of service attack (Active/Passive)
  + The user cannot perform any query on the database
  + However, this can be avoided using standard techniques of overcoming denial of service attack
* Authorization failure attack (Active) privilege
  + The user cannot perform any query on the database
  + However, the user is informed with standard error code in the feedback sent.
* Query malleability attack
  + The query service portal can perform query malleability (Active) privilege
  + However, this can be avoided by sending digital signature of the query with it.
* Query id malleability attack (Active) privilege
  + If Query id is modified, then it produces results, but the genuine user can’t fetch the result at the final step.
  + However, this can be restricted by including Institutional id, user id and job id in the digital signature computation by the query service client.
  + Thus, by checking the digital signature and if any discrepancy is found the query is dropped with standard message sending back to the user.
* Incorrect digital signature is obtained from query service client (Active) privilege
  + Service query portal discards the query if digital signature check fails
  + sends feedback to the user stating a standard error message representing digital signature check fail.

**Service Registry**

* Denial of service attack (Active/Passive)
  + User is not allowed to perform any query on the database
  + This can be avoided using standard techniques of overcoming denial of service attacks
* Service authentication failure attack (Active) privilege
  + Query discarded from further processing
  + However, the user is informed sending feedback with standard error message mentioning service authentication fail
* Service registry sends incorrect user metadata (Service Id, Service Name, HE Parameters, HE Client) (Active)
  + This kind of discrepancies can be identified by storing digital signature computed as DS(institutional id + user id + Service Id + Service Name + HE Parameters + HE Client )
  + Thus, when TEE gets the service parameters (Service Id, Service Name, HE Parameters, HE Client) from Service Registry and query information (institutional id, user id) from query it computed the signature and verifies that it has indeed received the correct service parameters.
  + If the computed DS do not match with that of the obtained DS, then he can perform either of the following two operation
* He can either ask for service parameters from service registry
* He can send feedback with stand error code representing service parameter check fail

**Service Key Management System**

* Denial of service attack (Active/Passive)
  + Query processing not allowed to perform on the database
  + This can be avoided using standard techniques of overcoming denial of service attack
* Key malleability attack (Active) privilege
  + This can be avoided by storing Public key and Evaluation key encrypted with HSM using symmetric key encryption like AES with its DS.
  + The digital Signature is computed as DS(institution id + user id + HSM(public key) + HSM(Evaluation key))
  + TEE performs DS check if inconsistency is observed

1. It either requests the service key again from Service key Management system or
2. Sends feedback to the user stating the standard error message.

* Wrong key Sent
  + As mentioned above DS can be used to check whether wrong key has been sent by the service key management system.

**Data Store**

* Denial of service attack (Active)
  + Query processing not allowed to perform on the database
  + This can be avoided by using standard techniques of overcoming denial of service attack
* Decryption failure due to wrong data provided for computation (Active) privilege
  + This type of attacks can be avoided by encrypting the database using the attestation key of TEE.

**FHE Query Engine**

* Denial of service attack (Active/Passive)
  + Query processing not allowed to perform on the database
  + This can be avoided by using standard techniques of overcoming DOS
* Decryption failure attack due to different circuit computation (Active) privilege
  + Query is feed along with its digital signature to the TEE
  + TEE computes digital signature of the query and checks to find out correct circuit has been provided for the computation
  + If incorrect circuit has been provided for computation, then sends feedback to the user with standard error message.
  + Thus, this type of attack can be avoided by computing and verifying digital signature inside TEE
* Query result malleability attack (Active) privilege
  + FHE Query Engine can modify the query result
  + However, this can be restricted by computing query result digital signature in the TEE itself before providing it to FHE Query Engine.
  + Send both encrypted query result and signature to the encrypted result store
* Side channel attack (Active/Passive)
  + Side channel attack can be avoided by computing query result in the TEE
* Known plaintext attack (computation performed in plaintext domain) (Active)
  + Same as that of the above i.e., computation performed in the TEE

**Encrypted Results Store**

* Denial of service attack (Active/Passive)
  + User is denied from fetching the query result.
  + However, this can be avoided using standard techniques of overcoming DOS
* Query result malleability attack (Active) privilege
  + TEE before decryption computes signature and checks its integrity with that of the signature received from the FHE Query Engine
  + If any discrepancy is observed, he sends feedback stating query integrity breached
* Decryption failure attack (Active) privilege
  + If decryption fail is obtained, then it must be genuine and, in such scenario, the obtained result is sent to the User.
* Side channel attack (Passive)
  + Decryption of the encrypted query result is performed in the TEE thus avoiding the side channel attack
  + Also, TEE directly send the decrypted query result to user using HTTPS connection avoiding further attacks while query result is in transit.

--------------------------------------------End---------------------------------------------

1. Attack to the HE system – Added more errors to the query to make it decryption failure, corrupted ciphertext, corrupted result, affective ways of communication, encoding is known to the attacker.

------Quantify the risk for all the above possibilities?

1. How the system behaves when one entity crashes?
2. Effect of behaviour of one component to the next component when they are in pipeline?
3. What happens when more than one entity comes together to perform an attack?
4. How to avoid Service Query Portal performing active attack say changes the digital signature of the user by its own signature? Can he perform such attack? May be restricted using centralized key storage
5. What will be the overhead of using TEE in the cloud computation as mentioned above in the encrypted cloud computation? Present in literature mentioned in ETH paper
6. What information is leaked by the feedback system? Done
7. Can a genuine user mount an attack to get secret information (plain data or the secret key)? Done (It can perform decryption failure attack adding more error in the query encryption phase)
8. What side information we can collect from the system? Done computations are performed in TEE. Thus, hopefully no information is leaked.
9. Attacks on API gateway? Done
10. Underline messaging system? Done
11. Corrupt the ciphertext in the client side that is not decryptable-correctly resulting into decryption failure? Already addressed using digital signature

Diagram

Description automatically generated

**When more than one entity come together to perform an attack**

**Client Service Portal and KMS**: When these two entities come together, they do not bring any new advantage to perform any new attacks in addition to those mentioned above.

**User and KMS**: When these two-entity come together for attacks they can perform all attacks mentioned above. In addition, the user now has advantage of access to all institution/user keys using these keys the user can ask services of different kinds. Such situations are normal when KMS gets compromised.

However, this problem can be avoided by storing keys encrypted using some attribute-based encryption scheme so that those keys can be decrypted for use by the authorized institution/user only.

**User, Client Service Portal and KMS:** Addition of Client Service Portal to the combination of User and KMS do not give any new advantage to lunch any new attacks other than the one mentioned above. When the entity User, Client Service Portal and KMS come together any unauthorized user can access the authorized user’s Institution/user keys and get services as a legitimate user without getting caught. However, this attack is same as that of combination of User and KMS and can be restricted by using the same technique as that of mentioned above.

**Service Query Portal and User Information Store**: When Service Query Portal and User Information Store gets combine, they cannot lunch any new attack. If they try to perform any malleability they will get caught as they must send a feedback message and query signature is centrally handled.

**Client Service Portal and Service Query Portal:** No new attacks can be lunched if these two entities combine. This is because if they change anything in the query, then either they need to send feedback to the user or they need to send the query to the next step where query signature is checked against the query. Thus, in either case they will be caught.

**User and User Information Store:** If the user is an authorized user, then this gives no advantage to the user. However, if the user is an unauthorized user, then this results in giving access to the authorized users information. However still the unauthorized user may not be able to ask any query if the Client Service Portal is 2 tier authentication system i.e., to login to the Client Service Portal requires both password and OTP. Thus, this gives overall no advantage to perform any new attack.

**KMS and User Information Store:** They can not lunch any new attack even if they collaborate. This is because they need to send feedback to the user if they perform anything that is not allowed to perform based on the protocol. If they do not follow the protocol, then they will get caught by the user or in the next step of the computation.

**User, Client Service Portal, KMS,** **Service Query Portal and User Information Store:** Collaboration of Service Query Portal and User Information Store to the User, Client Service Portal, KMS as mentioned above gives no new advantage as in the newly added entities only the authentication gets performed. As the user has access to the KMS he can use the genuine user’s information from KMS rather than using any unauthorized user information, so this added two entities gives no advantage than that of the combination of User, Client Service Portal and KMS.

**FHE Query Engine and Service Registry:** If they combine, then they can change the service type check however they can’t deny the query of an authentic user as if they do so then they must send feedback mentioning the reason for the deny and in that process, they will get caught. Also, they can’t perform any malleability attack on the query as if they do so then they will get caught in the query integrity verification phase performed in the TEE.

**User and Service Registry:** If user collaborates with the Service Registry, then he can perform any query request regardless of whether he has access to perform such query request or not and can get the service. It is hard to restrict such attacks viz. when service registry gets compromised.

**User, Client Service Portal, KMS, Service Query Portal, User Information Store, FHE Query Engine and Service Registry:** If all mentioned entities combine then any user can access the service using the genuine user’s credentials. However still the combination of the mentioned entities cannot get any information about the secret key or the database.

**User and Service KMS:** When the user has access to the KMS he will not get any new information from it as the KMS is encrypted with the attestation key of the TEE. Thus, no new attacks can be lunched.

**User and Database:** The database is encrypted with the attestation key of the TEE thus no new information can be gained, or no new attacks can be lunched.

All possible cases of combinations of different entities have been covered above and other than combination of USER and Service Registry in other cases no new attacks can be lunched.

**Behaviour of One component to the next Component:**

**Overhead of using TEE in the cloud computation in the Encrypted Domain**

Trusted execution environment (TEE) can be used for integrity protection. Attestation is used for integrity protection where the programs are executed in the server inside the enclave. However, the ciphertext are usually large in size and may result in performance bottleneck for computation as TEE are usually more restricted in terms of memory and available computational power then underlying untrusted computational environment. Recently in [1] this has been implemented and they presented some concrete numbers considering different circuits computed in the TEE. Here is the table taken from the paper [1] for the concrete numbers.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Program executed | Setup | Toy  Prover | Verifier | Setup | Small  Prover | Verifier | Setup | Medium  Prover | Verifier |
| FHE | .003 s | .002 s | .001 s | .807 s | .011 s | .009 s | 1.053 s | .014 s | .010 s |
| TEE | - | .154 s | - | - | 1.100 s | - | - | 1.260 s | - |

Table: Performance results for different instantiations of verifiable Fully Homomorphic Encryption. For FHE, Setup = Key Generation, Prover = Homomorphic Computation and Verifier = Encryption/ Decryption. Parameters used are *N=8196* and *log2 q= 137 = 45 +46+ 46*. (1) Toy circuit computes a ciphertext-ciphertext multiplication on two inputs provided by the client. (2) Small circuit computes low-depth two party computation, computing *x.v+w* for an encrypted client input x and a private server input v and w. (3) Medium circuit computes *ModSwitch((x-w)2)* for a client input x and server input w.

**Reference**

1. Verifiable Fully Homomorphic Encryption: Alexander Viand, Christian Knabenhans, Anwar Hithnawi

**Leakage**

Assumption: Whatever is known or inferred from the output of the computation is not considered as a leakage. Something known by an entity that is not meant to be known means leakage.

In the proposed FHE computation environment everything is computed inside TEE and before computation TEE checks genuineness of the data and circuit obtained for computation. Thus, even if any component tries to perform an active attack on FHE computation, he/she will be caught in the TEE checking phase performed before the circuit computation. Unless a genuine user acts as an attacker and uses a special circuit to extract secret information using the help of decryption failure.

Assumption: Service Query Portal cannot change signature of a query with another users/his own signature. In such a case even, the circuit can be changed by an active attacker and the TEE will assume that he is computing the correct circuit.

**Attack on API gateway**

This kind of attack can be avoided by using two-way authentication as mentioned below in the leakage section.

This can be avoided using secure channel communication. So that attacker cannot change anything when two parties communicate with each other.

**Query Service Client**

* The attacker may try to steal login information of user by impersonation attack (attempt to gain unauthorized access to systems by masquerading as authorized users)
* These types of attacks can be restricted by using additional layer of authentication in the form of OTP to use the facility.
* Eavesdrop the communication link, modify the communicated data or prevent the communication between the two
* These types of attack can be prevented using secure communication link between the communication parties.
* Attackers may use feedback system to steal login credential
  + Attacker may try to perform social engineering or brute force attack techniques to guess login credential of any authorized user in combination with feedback system.
  + However, as mentioned above the use of OTP based login nullifies these attack techniques

**Service Query Portal**

- Query well formedness is checked by computing

* Whether the query is correctly formed or not by checking query is encrypted by a valid user or not, metadata used in the query is correct or not, service parameters are correct or not etc.
* However how to check whether the user has used more error to form a query to make decryption failure type of attack.
* Knows query frequency of institution/user
* Use feedback to extract secret information (Leakage by feedback message)
* Feedback messages send from service query portal to the user is of authentication failed message. The user whoever encrypted the query must be a genuine user as he has to login to the client service portal to form a query. Thus, the user who ever successfully encrypts the query is a genuine user, so feedback reveals no extra information to the genuine user.
* Additionally, the user comes to know by the feedback message if any active attack is going on the service side.

**Service Registry**

- Leakage of feedback message

* Feedback messages send from service registry to the user is of authentication failed message. The user whoever encrypted the query must be a genuine user as he has to login to the client service portal to form a query. So, feedback message reveals no extra information to the genuine user.

- Knows Query frequency of institution/user

**Service Key Management System**

- Leakage of feedback message

* Same as that of above

- Knows Query frequency of institution/user

**FHE Query Engine**

- Leakage of feedback message

* Same as that of above

- Knows Query frequency of institution/user

**Encrypted Result Store**

* Knows query frequency of institution/user

--------------------------------------------End---------------------------------------------

**Different types of attacks possible to each component**

Query service client

* Denial of service attack (Active/Passive)
* Malleability of query attack (Active) privilege

Service Query Portal

* Denial of service attack (Active/Passive)
* Authorization failure attack (Active) privilege
* Query malleability attack resulting into corrupted ciphertexts (Active) privilege
* Query id malleability attack resulting attacker into the system (Active) privilege

Service registry

* Denial of service attack (Active/Passive)
* Metadata malleability attack resulting into decryption failure attack (Active) privilege
* Authorization failure attack (Active) privilege

Service KMS

* Denial of service attack (Active/Passive)
* Metadata malleability attack resulting into corrupted result (Active) privilege

Data Store

* Denial of service attack (Active)
* Decryption failure attack due to wrong data provided for computation (Active) privilege

FHE Query Engine

* Denial of service attack (Active/Passive)
* Decryption failure attack due to different circuit computation rather

then the one requested (Active) privilege

* Ciphertext malleability attack (Active) privilege
* Side channel attack (Active/Passive)
* Known plaintext attack (computation performed in plaintext domain) (Active)

Encrypted Results Store

* Denial of service attack (Active/Passive)
* Side channel attack (Passive)
* Ciphertext malleability attack (Active) privilege
* Decryption failure attack (Active) privilege

**Mitigation of Attacks mentioned above**

Query service client

* Denial of service attack

1. This attack can be avoided using the traditional techniques of avoiding the Denial-of-service attack.

* Malleability of query predicates

1. This attack can be avoided using MAC with the query.
2. If the received MAC is same as that of the MAC of the query, then process it else discard the query.
3. Send the MAC to the next step of computation.
4. If the MAC check fails send feedback to the query user stating MAC check failed.

Service Query Portal

* Denial of service attack

1. Same as that of mentioned above

* Authorization failure attack

1. This active attack can be avoided using feedback to the query user mentioning the reason behind the authorization fail.

* Query malleability attack resulting into corrupted ciphertext

1. This attack can be avoided using MAC of query along with the query
2. If the received MAC is same as the of the MAC computed using the query, then only process the query else discard the query.
3. If the MAC check fails send feedback to the query user stating MAC check failed.
4. If MAC check is done successfully then Send the query and its MAC to the next step.

* Query id malleability attack

1. MAC restricts such kind of attacks.

Service Registry

* Denial of service attack

1. Same as that of mentioned above

* Metadata malleability attack resulting into decryption failure attack
* Authorization fails attack

1. This attack can be avoided using feedback to the query user mentioning the reason behind the authorization fail.

Service KMS

* Denial of service attack

1. Same as that of mentioned above

* Metadata malleability attack resulting into decryption failure attack

Data Store

* Denial of service attack

1. Same as that of the mentioned above

* Decryption failure attack due to wrong data provided for computation

1. I don’t have any idea as of now how to avoid this.

FHE Query Engine

* Denial of service attack

1. Same as that of mentioned above.

* Decryption failure attack due to different circuit computation rather than the one requested

1. I don’t have any idea as of now how to avoid this.

* Side channel attack

1. Can be avoided using SGX kind of secure computing environment for the computation.

* Known plaintext attack (computation performed in plaintext domain)

1. May be avoided using SGX kind of secure computing environment for the computation.

Encrypted Results Store

* Denial of service attack

1. Same as that of the mentioned above

* Side channel attack

1. Can be avoided using SGX type of secure computation environment.

* Ciphertext result malleability attack

1. Can be avoided by adding MAC of the ciphertext result. Adding of the MAC can be done by the FHE Query Engine.
2. Send both computed ciphertext result and MAC to the Encrypted result Store.

* Decryption failure attack

1. Use of MAC along with the computed ciphertext result can restrict this attacks.

**Data Classification**

Data is classified into 3 categories

1. Extremely Private data

* Secret key used for encryption
* Query data of the user
* Database used for computation

1. Normal private data and

* Institutional id/User id

1. Public Data

* Everything else is public data

**Attacks Classifications**

Ciphertext verification attacks

Passive attack by different components

Active attack by different components

Denial of service attack

Secret key recovery attack

Side channel attack

Brute force attack

Ciphertext-only attack

Chosen plaintext attack

Chosen ciphertext attack

Known plaintext attack

Key and algorithm attack

Replay attacks

Linear cryptanalysis

Man in the Middle Attack

Adaptive chosen-plaintext attack