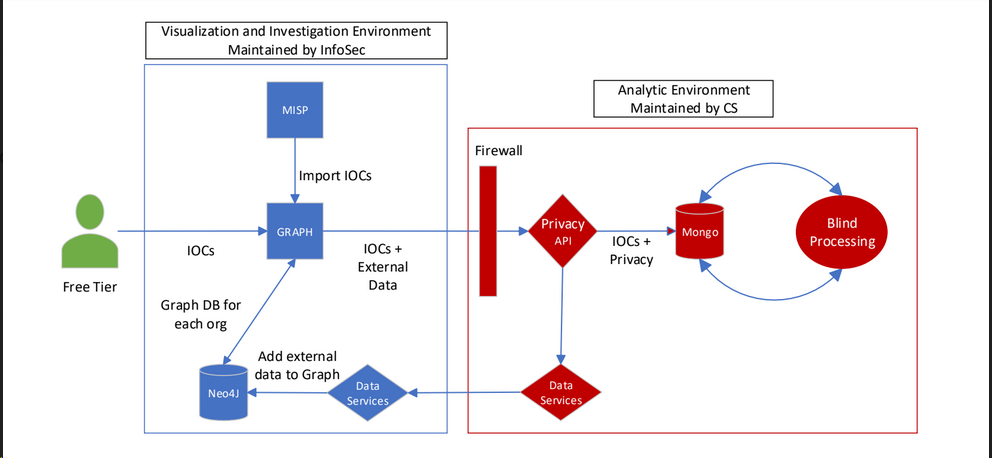
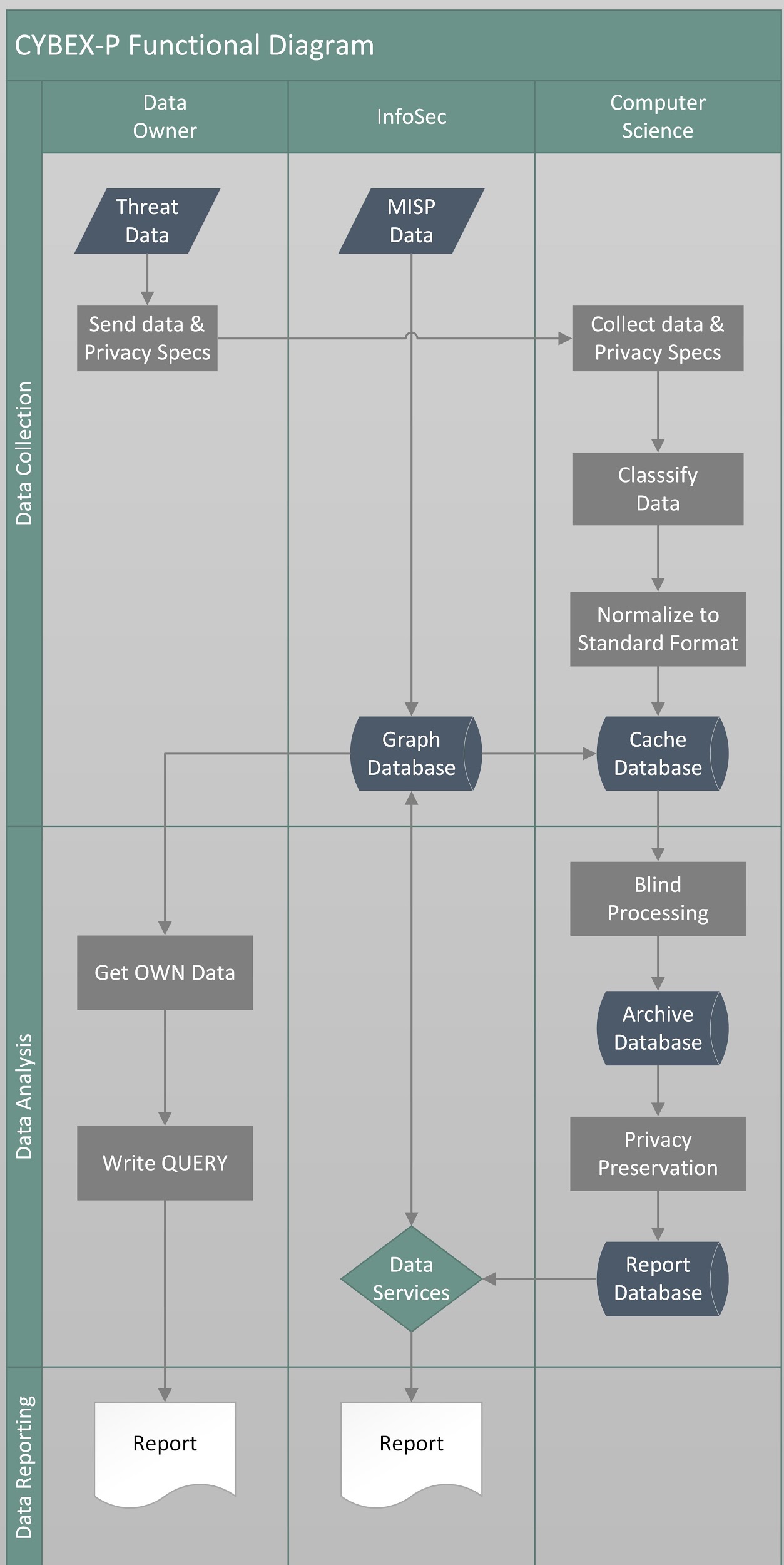
**Functional Overview of CYBEX-P**

**(This draft for and will undergo significant modifications)**

**(Edit request: Farhan)**

****

****

Principal functions of CYBEX-P system are:

1. **Data Access includes**
   1. Data Collection:
      1. Getting data into CYBEX-P in a secured manner
         1. Web-interface for organizations to upload data
         2. Automated scripts to get real-time data
      2. Storing data in a secured manner
      3. Parsing data into standard format (STIX 2.0)
   2. Data Reporting
      1. Creating graphs out of data
      2. Graph interface for custom queries

1. **Data Analysis** (both functions and features)
   1. Blind Processing - ensure no person (!) gets access to unencrypted data
   2. Privacy Preservation - removing sensitive information from data
   3. TPM - secured storage for encryption keys
   4. Correlation, aggregation, analysis etc.

Parties Involved:

1. Data owner:
2. Uploads/shares data with Infosec
3. Writes custom queries on own data to create meaningful report
4. Note: anyone with data can be owner. For example Bryson at the moment is the owner of the honeypot data at present.

2. Infosec:

1. Provides infrastructure for data access (data collection & data reporting)

3. Computer Science (CS)

1. Provides infrastructure for data analysis (blind processing, privacy preservation, tpm)

Data Sources:

1. MISP Data - owned by Infosec
2. All other data - honeypot, cuckoo reports etc.- owned by data supplier

Process Overview:

**Data Collection:** data can be collected in various ways:

1. Uploaded as file into web front end
2. Collected real-time via websocket
3. Parsed from files
4. Called using API
5. Shared via USB storage
6. ...
7. **Send data & Privacy Specs:**
   1. Data owner will share data with Infosec in agreed upon method (websocket, API, USB, shared file etc.).
   2. Data owner also must share privacy specifications of each data key in the following format:
      1. Level 0: Is not privacy sensitive at all and can be left unencrypted
      2. Level 1: Is privacy sensitive and must be encrypted by CYBEX-P. Symmetric key from TPM is used by CYBEX-P. CYBEX-P blind processing server has access to unencrypted data. No human has access to unencrypted data.
      3. Level 2: Encrypted by data owner and not even CYBEX-P has access to it. Data owner can give access rights to any participant (including other organizations). Uses PKI.
      4. Level 3: Data owner only wants to find out if others received similar data and no-one has access to unencrypted data.

Example of sending data and privacy specifications:

Data:

{

“name” : “John Doe”,

“email\_address”: “[johndoe@example.com](mailto:johndoe@example.com)”,

“email\_body”: “You have won USD 10,000.”,

}

Privacy Specification:

{

“name” : 0,

“email\_address” : 1,

“email\_body” : 3,

}

Explanation of data and privacy specification:

* The data has three keys - “name”, “email\_address” and “email\_body”. Keys are never encrypted and left as plain text
* The privacy specification for “name” is 0. So, this data is left as plain text and not encrypted and available for everyone (administrators, analysts) to see.
* The key “email\_address” has a privacy specification of 1. So, CYBEX-P encrypts it with own symmetric key. Only blind processing unit has access to unencrypted value. No human can read the plain text value.
* The key “email\_body” has privacy specification of 3. Which, means the Data owner wants to know if the same email was received by any other person in any other organization. Private Set Intersection is used for it. No human or machine can read the unencrypted/plain text body of the email.

1. **Collect data and privacy specs:**
   1. Infosec collects data and privacy specs from data owner in agreed upon method.
2. **Classify Data:**
   1. Infosec classifies data according to the type of data received.
3. **Normalize into standard format:**
   1. Infosec parses data into standard format (STIX 2.0 or any other suitable format) and make the data structure available as a separate document.
4. **Store data into Graph Database:**
   1. Infosec stores data in their graph database for future processing
5. **Store data into Cache Database:**
   1. Infosec transfers data from graph database to cache database using API provided by Computer Science (CS)

**Data Analysis (Technical in Appendix I):** data is analyzed in two separate ways

1. Data owner has access to their own data. Data owner can save a custom query in neo4j that produces a report out of their own data. This data is publicly available to everyone.
2. Computer Science (CS) has access to all data and create results based on privacy requirements. (Data with privacy level 0 can be processed by anyone, data with privacy level 1 must be processed using blind processing module without human interaction.
3. Data analysis is a continuous processing and will carry forward in iterations.
4. Any data in the archive database can be further input into the processing pipeline

Example of iterative data analysis:

Example data in archive database, we will call it **D0**:

**D0:**

{

“type” : “x-honeypot”,

“Src\_ref” : “1.1.1.1”,

“event\_id”: “cowrie.file.download”,

“msg”: “File with hash 28376ABCD downloaded from http://www.example.com”

}

Lets pass it through three data analysis filters: **F1, F2, F3** simultaneously:

**D0** produces **D1, D2, D3** when passed through **F1, F1** and **F2** respectively.

**F1** extracts the url and produces **D1** as follows:

**D1:**

{

“type” : “x-malicious-url”,

“value” : “http://www.example.com”,

}

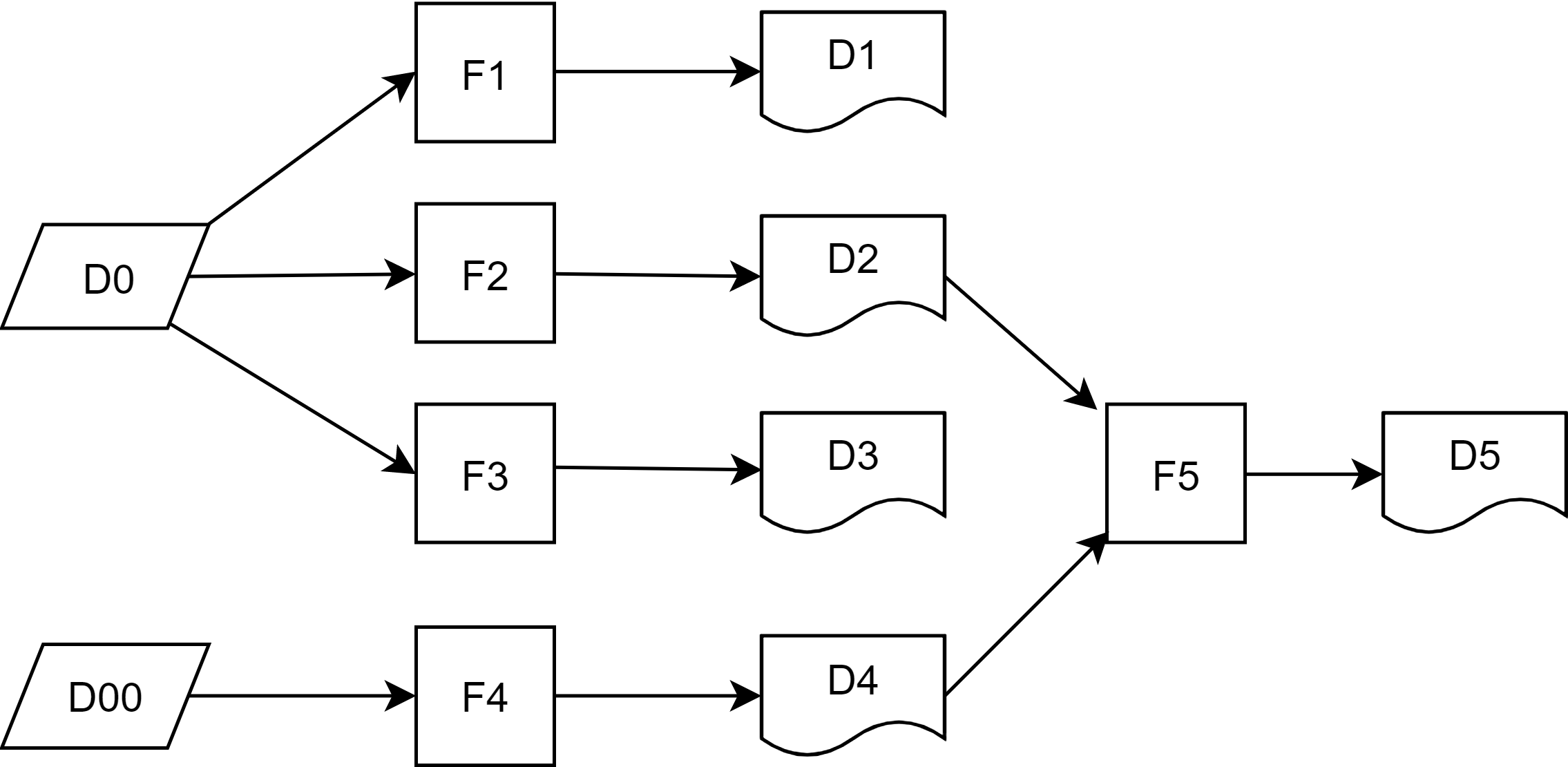


Figure: Iterative Data Analysis

**F2** extracts the attackers source ip address and produces **D2** as follows:

**D2:**

{

“type” : “x-malicious-ipv4-address”,

“value” : “1.1.1.1”,

}

**F3** extracts the malicious files hash and produces **D3** as follows:

**D3:**

{

“type” : “x-malicious-file”,

“hash\_alue” : “28376ABCD”,

}

Now let us consider another event data called **D00** represents a network scan:

**D00:**

{

“type” : “x-network-scan”,

“src\_ref” : “2.2.2.2”,

“Description” : “Scanned local server for open ports”

}

**F4** extracts the attackers source ip address from **D00** and produces **D4** as follows:

**D4:**

{

“type” : “x-malicious-ipv4-address”,

“value” : “2.2.2.2”,

}

**F4** and **F2** both extract malicious *ip-address* from another data but they are different analysis filters because the source data structure (**D2** and **D00** here) are different.

Now since both **D2** and **D4** both are of same data type we can pass them through the same filter called **F5** which creates an aggregate list **D5** of malicious ip addresses called *ip-address-blacklist*

**D5:**

{

“type” : “ip-address-blacklist”,

“value” : [“1.1.1.1”, “2.2.2.2”],

}

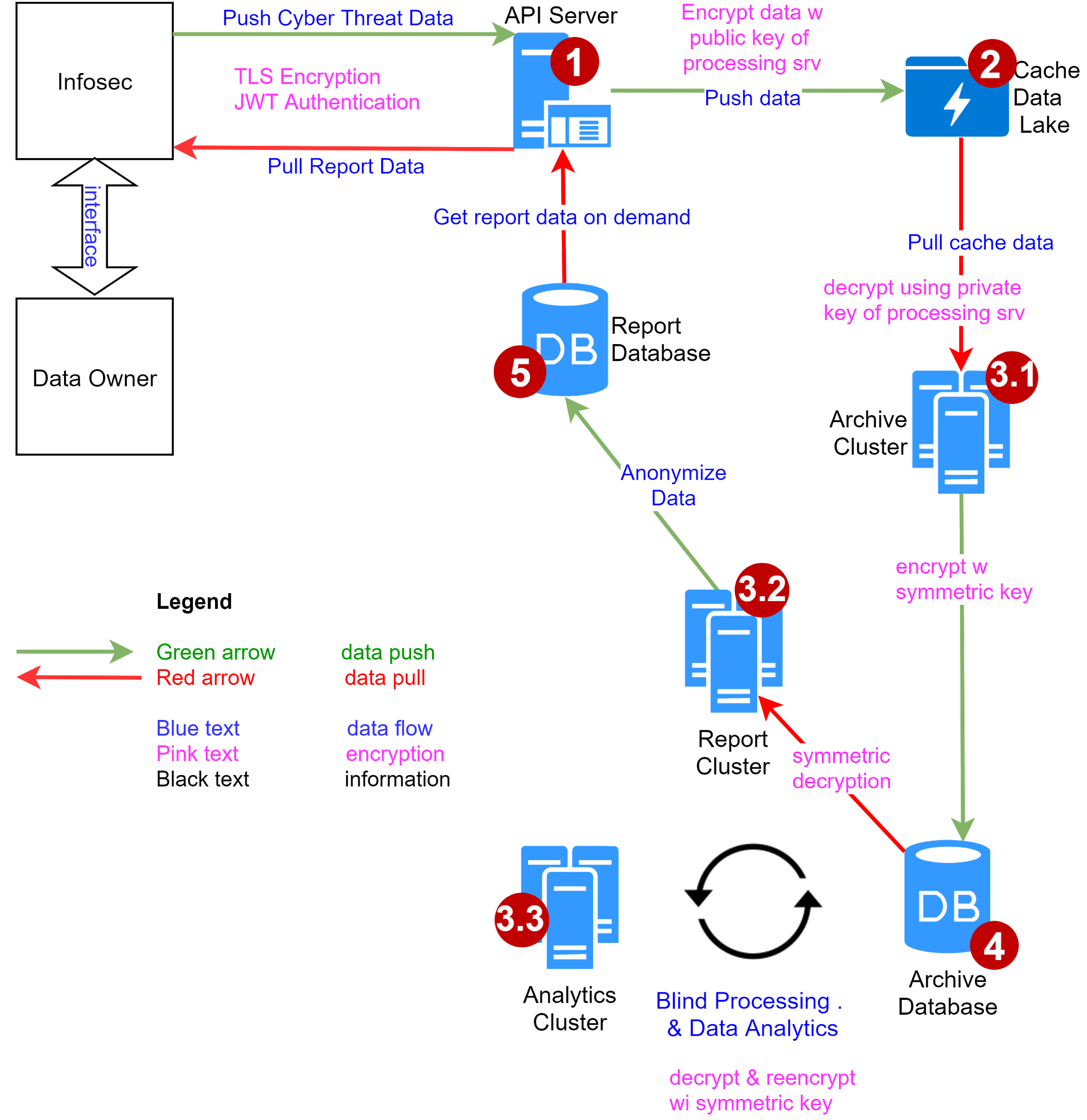
1. **Blind processing**
   1. Computer Science (CS) analyses data using blind processing module
2. **Store data in report database:**
   1. CS stores the processed data in archive database
3. **Privacy Preservation**
   1. CS performs a second round of privacy preservation on the data by removing organization id and others
4. **Store data in report database:**
   1. CS stores the data in report database
5. **Store data in graph database:**
   1. CS sends privacy preserved data to Infosec for storing in graph database and reporting

**Data Reporting:** reports can be shown in several ways.

1. Data owner creates custom query in neo4j that is shown as dynamic report for everyone
2. Infosec shows data as received from CS as static reports
3. Infosec creates own reports out of data from graph database

**Integration with existing project:**

Following diagram outlines an example how the new operations of CYBEX-P can be integrated with the existing ones.



For the purpose of this diagram Infosec is considered as a black box that interacts with both Data owner and CS.

*Background*

* There are 3 servers (virtual)-

1. Machine 1 = WEB/API server [FQDN = cybexp1.acs.unr.edu]
2. Machine 2 = Processing server [FQDN = cybexp2.acs.unr.edu]
3. Machine 3 = Database server [FQDN = cybexp3.acs.unr.edu]

*1* - *API Server (Farhan)*

* Has two purposes - POST and GET
* POST Method - Input plain text file, stores encrypted text file in cache DB6 
  + Encrypts file using AES and random key.
  + Encrypts the random key using public key of processing server
  + Puts the encrypted key and the encrypted file in cache db
* GET Method - Returns data from report DB10
  + Infosec collects this data for reporing
* Uses JWT authentication
* Built with Django rest framework.
* Codes are in machine 1 (WEB/API)

*2* - *Cache Data Lake (Khalid)*

* MongoDB instance in DMZ
* Stores encrypted text files with raw threat data as received from client
* Files are encrypted by AES using random key. The random key is encrypted with public key of processing server and attached to the beginning of each file.
* Database presently in machine 3 (database server) but see future plan below
* Future plan - get separate machine in DMZ as shown in diagram (Khalid)

*3 - Processing Server (Not in diagram)*

* The processing server is drawn as three blocks in diagram - 3.1, 3.2, 3.3 because the three processes are very different
* The server is the machine 2 as described in the beginning
* Future plan - make it a TPM = Trusted Platform Module (Batyr??)

*3.1* - *Archive Cluster (Khalid, Farhan)*

* Not really a cluster yet; a Java code in machine 2 (Processing Server)
* Java code manages the flow with inside calls to several python scripts
* Decrypts unprocessed file in cache db -> converts it into STIX2 -> re-encrypts with AES -> stores in archive db8
* Codes are in machine 2 (Processing Server)

*3.2 & 3.3 are described later to be coherent with data flow*

*4 - Archive Database (Khalid)*

* Encrypted database inside of the firewall
* Data is in STIX2 (or any standard?) format.
* DB in machine 3 (Database server)

*4.3* - *Analytics Cluster (Jay, Farhan, Khalid)*

* Heart of CYBEX-P & research aspect
* Performs advanced analytics on STIX2 data in archive DB8
* Privacy preservation can be done in this step as reports are generated or in the next step (7.2) as reports are sent to the dashboard
* Expected process flow is as follows

Reads encrypted data from archive db8 ⇒ ⇒ Decrypts data in archive db8 ⇒ ⇒ performs analysis on data to create reports (could be privacy preserving) ⇒⇒ re-encrypts data and stores back in archive db8 / sends it to report db

* Scripts are in machine 2 (Processing server)

*4.2 Report Cluster (Farhan/Khalid??)*

* As of the now, the plan is to use this part of the processing server to decrypt data in archive db, anonymize it and then send it report db9.

*5*- *Report Database (Khalid)*

* Database outside of the firewall that holds categorized and more polished, simpler, analysed data.
* Data is anonymized and contains no user information

**Integration with existing project:**

Following diagram outlines