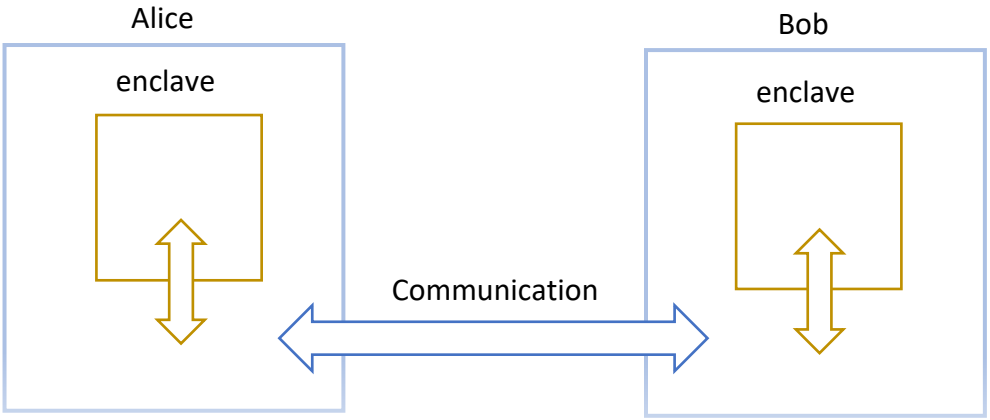
Secure Function Evaluation -with SGX

Design:

המערכת שלנו מורכבת משתי אפליקציות Alice ו-Bob אשר כל אחת מהם משתמשת בטכנולוגית intel SGX ליצור סביבת הרצה מאובטחת (TEE), ולכן כל אפליקציה משתמשת ב-enclave שמספק לה את זה.



Communication:

האפליקציה וה-enclave מתקשרים בעזרת ה-Ecalls וה-Ocalls , בהנחה ששתי האפליקציות רצות על אותה מכונה התקשורת ביניהם מתבצע בעזרת files, אחרת היינו צריכים לבצע את זה באמצעות sockets.

Enclave Structure:

- private/public keys לחתימה.
- מפתח הצפנה סימטרי.
- map של הלקוחות וההוצאות שלהם.

ה-enclave שומר ה private/public keys שלו בקבצים:


Public key


Sealed private key

ה-enclave משתמש בקבצים אלה ל-IPC:


Public DH key


Encrypted clients

Process steps:

1. מתבצע אתחול של ה-enclave, אם זאת הפעם הראשונה שהוא עולה אז הוא יוצר מפתחות החתימה ושומר את הפרטי מוצפן ב disk באמצעות sealing, אשר רק הוא יוכל בעתיד לפענח אותו ולעשות לו unsealing, אחרת אם זאת לא ההרצה הראשונה הוא מנסה לשחזר מפתחות החתימה שלו ששמר קודם.

2. האפליקציה שולחת ל-enclave את קובץ הלקוחות, אשר הוא שומר אותם ב-map אצלו.

3. ה-enclave מתחיל תהליך הסכמה על מפתח משותף עם ה-enclave השני באמצעות DH.
   3.1. הוא יוצר מפתחות DH ומעביר המפתח הפומבי חתום.
   3.2. לוקח המפתח הפומבי של הצד השני ומוודא חתימה עליו.
   3.3. יוצר מפתח DH משותף DH_key.
   3.4. יוצר מפתח הצפנה משותף key_session = H(DH_key).

4. ה-enclave מצפין בעזרת key_session את הלקוחות שלו ושולח אותם חתומים לצד השני.

5. ה-enclave מקבל את הלקוחות של הצד השני.
   5.1. מוודה חתימה עליהם.
   5.2. מפענח אותם ושומר ב-map אחר.

6. חישוב הפונקציה הדרושה ע״י ה-enclave, והחזרת התשובה לאפליקציה.

Security implementations:

- מכיוון שלא סומכים על האפליקציה כל מידע שיוצא מה-enclave נחתם באמצעות המפתח הפרטי שלו אשר רק הוא יודע אותו, ולכן גם כל מידע שהוא מקבל הוא את בודק חתימה עליו.

- ה-enclave עושה seal לנתונים כך שרק הוא יוכל לבצע להם unseal. וכך הוא שומר המפתח הפרטי שלו מוצפן על הדיסק.

- השתמשתי בהצפנה סימטרית aes_ctr_128 עם מפתח ו-iv שנגזרו ממפתח DH שהוסכם בין שני הצדדים.

Concerns:

- ה-enclave לוקח את מפתח הפומבי של ה-enclave השני מהאפליקציה, ולכן מניחים שאנו סומכים על האפליקציה בשלב הזה, אבל כדי שזה יהיה יותר בטוח נדרש להעביר את המפתח הפומבי הצורה בטוחה או להעביר אותו ב-certificate.

- לפני ששני ה-enclaves ידברו אחד עם השני הם צריכים לבצע local attestation אבל לא עשינו את זה כאן לצורך פשטות, ואם שני ה-enclaves נמצאים על מכונות שונות הם צריכים לבצע remote attestation.

- כרגע יש לנו אותו enclave בשתי האפליקציות לכן שניהם יכולים לבצע unseal לנתוני אחד של השני.

Ecalls function:

```
/* ECALLS functions */
int ecall_init_signing_keys(const char* prv_file, const char* pub_file, const char* other_pub_file)

int ecall_process_clients_file(const char* file_name,const size_t lines_num)

int ecall_create_DH_keys(const char* my_key_file, const char* other_key_file)

int ecall_encrypt_clients_data(const char* target_file)

int ecall_decrypt_and_process_file(const char* target_file)

int ecall_calc_function()
```

Ocall functions:

```
/* OCall functions */
void ocall_printf(const char *str)

int ocall_wait_for_ready_file(const char* file_name)

int ocall_get_file_size(const char* file_name)

int ocall_save_file_data(const char* file_name, const uint8_t* data, const size_t len)

int ocall_load_file_data(const char* file_name, uint8_t* data, const size_t len)

int ocall_read_line_data(const char* file_name, int* a, int* b, const size_t len)

int ocall_remove_shared_file(const char* file_name)

int ocall_function_res(int avg)
```

לגבי הגדרות ה-enclave הגדלתי קצת את גודל המחסנית שלו כדי להעביר את תוכן קובץ הלקוחות באמצעות ocall_load_file_data.

לצורך כתיבת וקריאה נתונים לקובץ, בין אם זה מפתח פומבי ל-DH או הלקוחות, יש פונקציות עזר שעושות serialization ו-deserialization ממבני נתונים מסוים ל-byte stream ולהפך.

```
/* Serialization helpers*/
int serialize_key(const sgx_ec256_public_t key, uint8_t* buf)

int deserialize_key(sgx_ec256_public_t &key, const uint8_t* buf)

int serialize_map(const map<unsigned int,unsigned int> &mp, uint8_t* stream)

int deserialize_map(map<unsigned int, unsigned int> &mp, const uint8_t* stream, size_t len)
```

SDK function:

1. לצורך יצירת מפתחות חתימה השתמשתי בפונקציה : sgx_ecc256_create_key_pair שיוצרת מפתח פרטי ופומבי.

2. לצורך אחסון מאובטח של המפתחות השתמשתי בפונקציות: sgx_seal_data ו-sgx_unseal_data, שמצפינה המידע בעזרת מפתחות שידועות רק ל-CPU.

3. לצורך הסכמה על מפתח משותף השתמשתי בפונקציה:
sgx_ecc256_compute_shared_dhkey שלוקחת המפתח הפרטי שלי והמפתח הפומבי של הצד השני ויוצרת מפתח משותף.

4. לצורך יצירת מפתח להצפנה סימטרית השתמשתי בפונקציה: sgx_sha256_update ו-sgx_sha256_get_hash שמתמצת המפתח בעל 256 ביט למפתח ו-iv בעלי 128 ביט.

5. לצורך הצפנה סימטרית באמצעות aes_ctr השתמשתי בפונקציות: sgx_aes_ctr_encrypt ו- sgx_aes_ctr_decrypt שלוקחות מידע ומצפינות אותו בעזרת המפתחות הקודמים.

6. לצורך חתימה ואימות מידע השתמשתי בפונקציות : sgx_ecdsa_sign ו- sgx_ecdsa_verify, אשר בחתימה היא משתמשת במפתח הפרטי שלי, ובאימות משתמשת במפתח בפומבי של השולח.