Evaluation of a privacy-preserving embedded languages for Intel SGX

Eloi Besnard, Joshua Randria

Gaël Thomas, Subashiny Tanigassalame, Yohan Pipereau

02/2023

Table of content

IntelSGX

Intel SGX ?

- Software Guard eXtension
- Zones mémoires protégées, chiffrées par le CPU (**Enclave**)
- Secure web browsing, secure remote computation, ...

Privagic

Comment utiliser Intel SGX ?

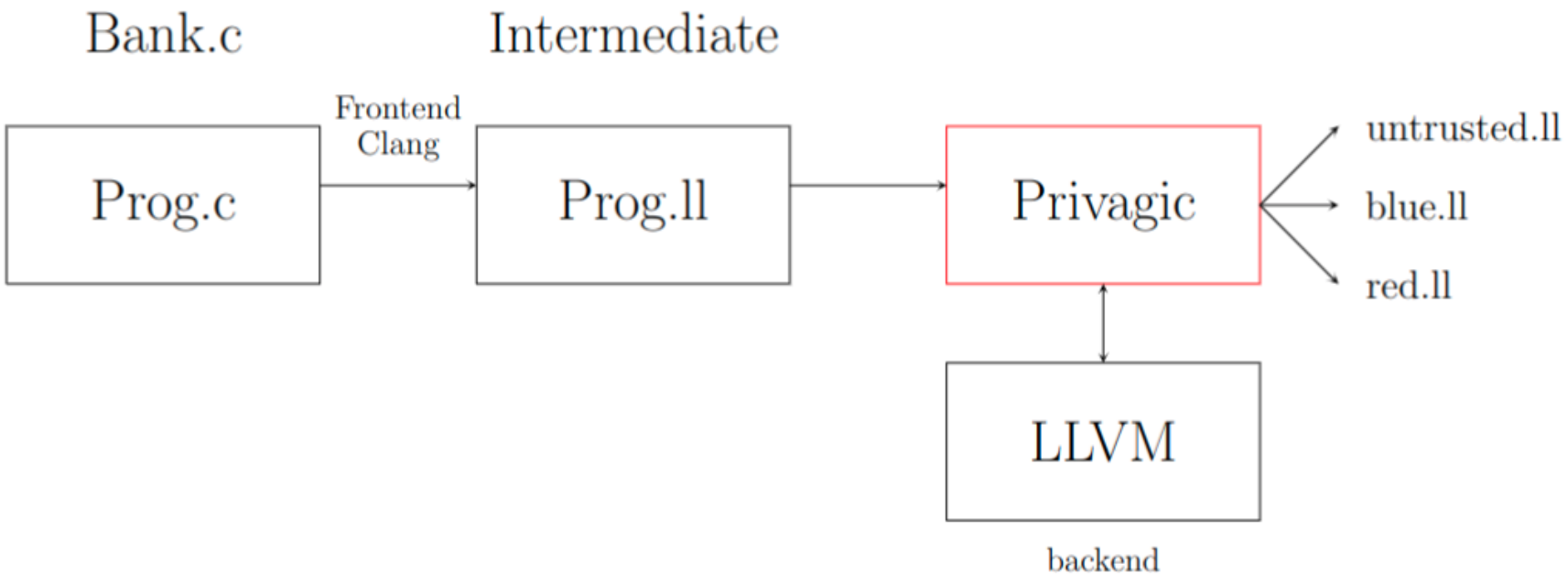
- Manuellement séparer les données
- MAIS: Perte de temps!
- MAIS: Susceptible de faire des erreurs!

Solution: **Privagic**

- Privacy-Magic
- Surcouche du compilateur LLVM sachant lire des annotations pour séparer le code automatiquement

Illustration Privagic

```
1 struct bank_account {
2     char* name[256] color(red);
3     int val color(blue);
4 }
```



Objectifs

But: Evaluation des performances de Privagic à l'aide de deux implémentations d'une banque répartie

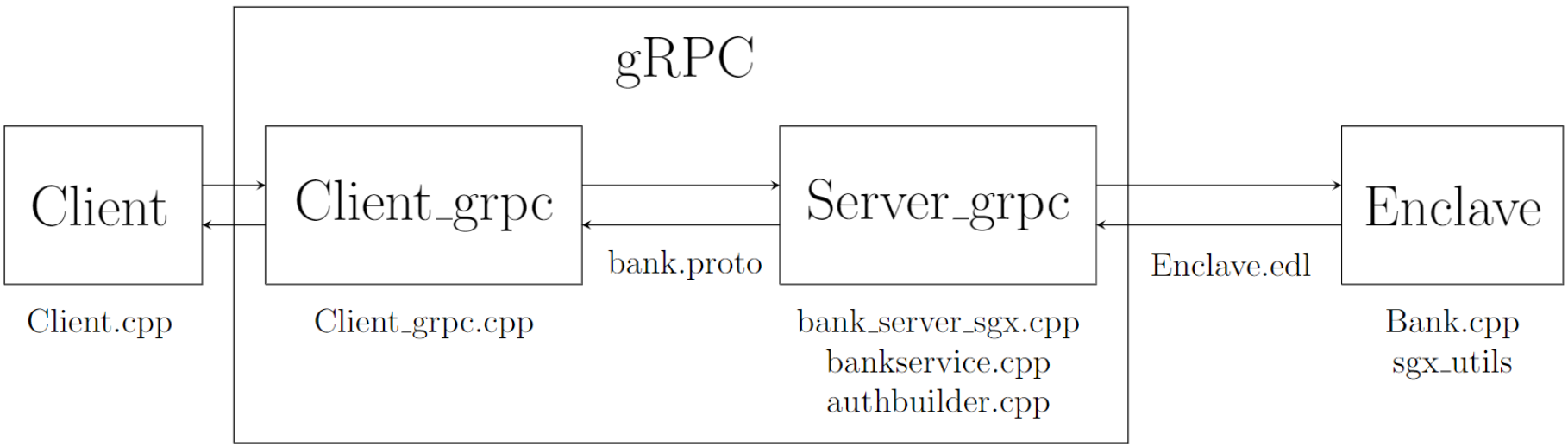
Objectif 1: Implémentations d'une banque répartie

- 1. Bank gRPC
- 2. Bank gRPC + SGX

Objectif 2: Evaluation

- Injecteur de charge
- Graphiques débit et latence

Architecture Bank



Architecture globale

Architecture Bank

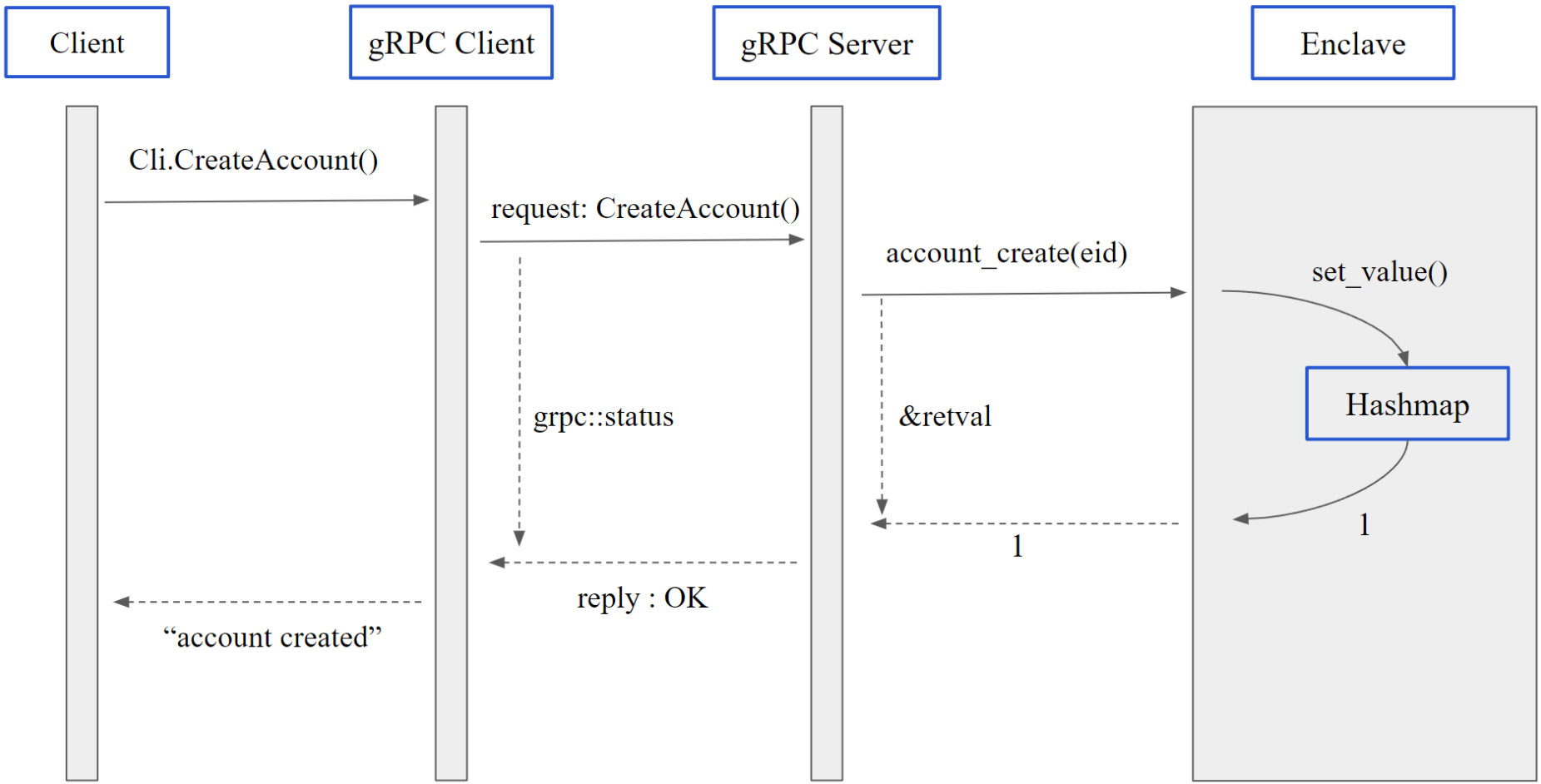


Diagramme de séquence

Etapes

Etapes

1. Code avec **Hashmap**
   hm: ⊘, gRPC: ⊘, SGX: ⊘, MKFL:✔
2. Code avec **gRPC**
   hm: ✔, **gRPC**: ✔, SGX: ⊘, MKFL:✔
3. Code avec **SGX**
   hm: ✔, gRPC: ⊘, **SGX**: ✔, MKFL:✔
4. **Fusion** SGX et gRPC
   hm: ✔, **gRPC**: ✔, **SGX**: ✔, MKFL:⊘
5. **Makefile** SGX gRPC
   hm: ✔, gRPC: ✔, SGX: ✔, **MKFL**:✔

gRPC

Google Remote Procedure Call

- Framework qui gère les requêtes et réponses entre server et client
- Bank.proto

gRPC:proto

service

- service: `rpc nom_methode(nom_methodeReq) returns (nom_methodeResp)`

```
service bank {
    rpc CreateAccount(AccountCreateReq) returns (AccountCreateResp);
    rpc AddAmount(AmountAddReq) returns (AmountAddResp);
    rpc SubAmount(AmountSubReq) returns (AmountSubResp);
    rpc ListAccount(Empty) returns (Empty);
}
```

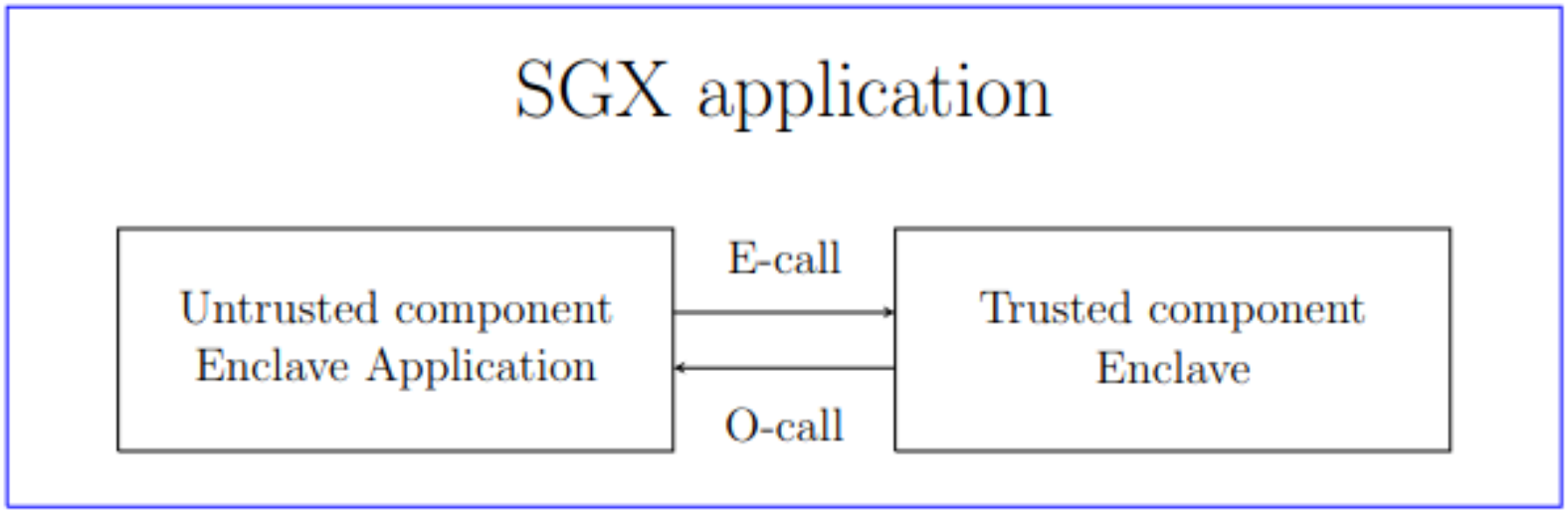
message

- message: Req ou Resp

```
message AmountAddReq {
    string account_name = 1;
    int64 amount = 2;
}

message AmountAddResp {
    int64 new_amount = 1;
}
```

Untrusted vs Trusted



SGX application with enclave

EDL

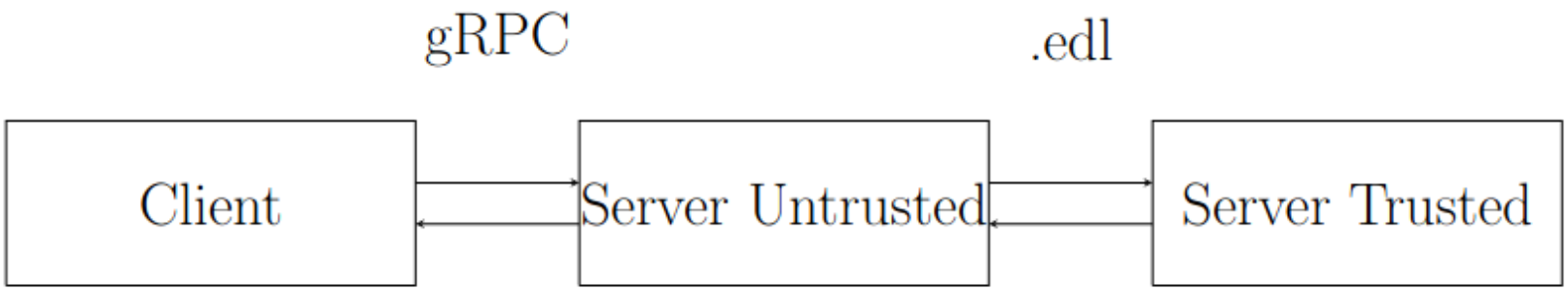
.edl

- .edl = Enclave Definition Language
- similaire au `Bank.proto`
- `Bank_create` en +

```
enclave {
        include "Bank.h"
    trusted {
        public int bank_create([in] int* capacity);
        public int account_create([in, string] const char* account_name);
        public int amount_add([in, string] const char* account_name, [in] int* amount);
        public int amount_sub([in, string] const char* account_name, [in] int* amount);
        public int list_accounts();
    };
    untrusted {
        void ocall_print([in, string]const char* str);
    };
};
```

Enclave.edl

Relation proto / edl



SGX-gRPC

Hashmap

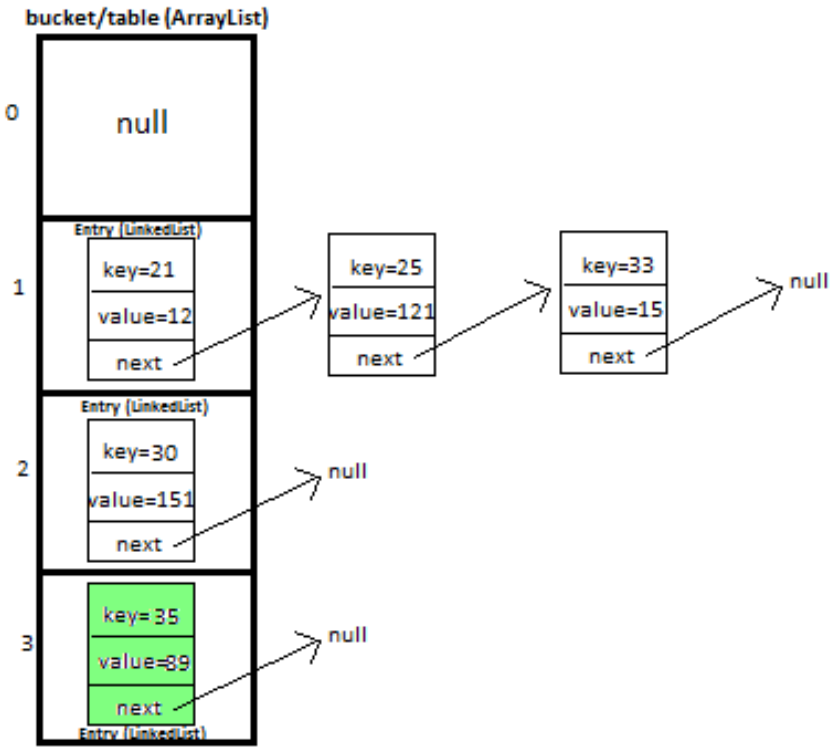
But

- Créer une structure de données pour la bank
- Assez simple pour passer de version base $\rightarrow$ version SGX
- Pouvoir choisir la capacité

Hashmap Illustration

Caractéristiques

- Table de hachage
- Hash function
- Liste chaînée



Makefile

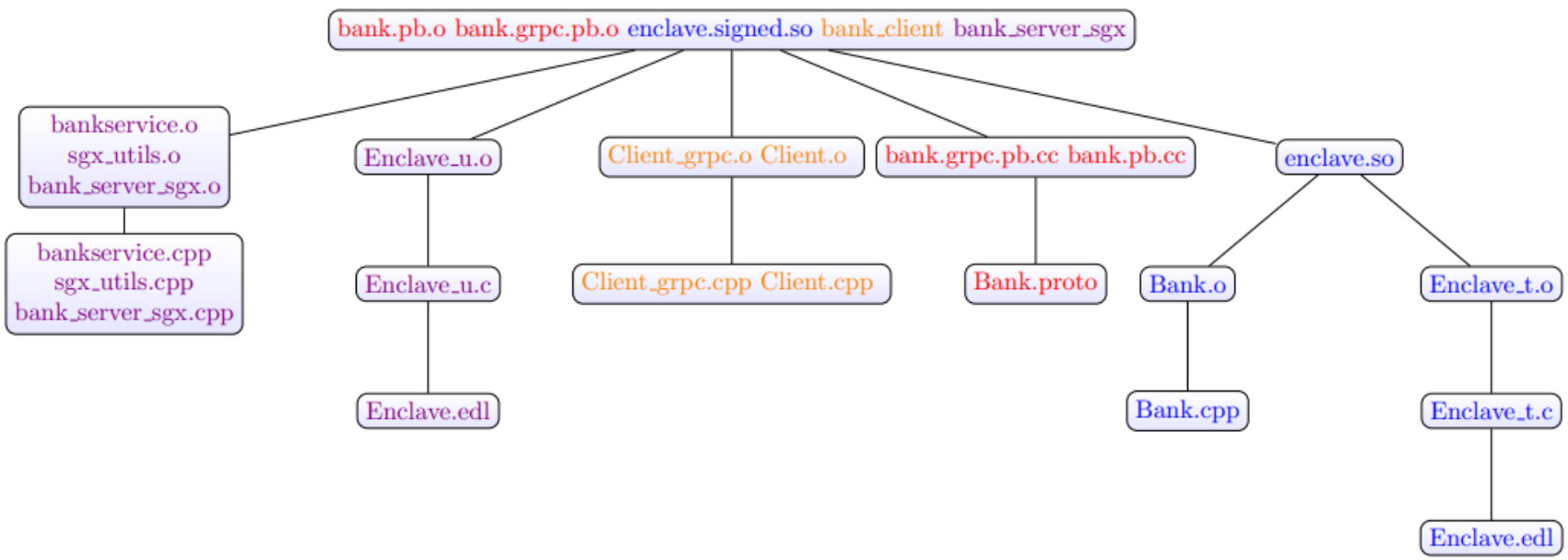


Figure: Makefile

Injecteur de Charge

But: Montrer que notre bank respecte la spécification de TPC-B

Client.cpp

Boucle de création et modification des comptes (*10 000)

- createAccount
- addAmount & subAmount

→ Mesures débit et latence

Graphiques

Ecriture des csv: `fprintf`

- Throughput : nbr d'opération (add &sub) / durée
- Latency : durée d'une opération

→ Lecture des dataframes : script python et blibliothèque pandas

throughputope.csv

```
1  i,throughputope,time
2  100,3.859561,0.025911
3  200,3.892541,0.051605
4  300,3.809458,0.077857
5  400,3.678909,0.105042
6  500,3.433701,0.134168
```

n°ope ope/ms s

latencyope.csv

```
1  i,latencyope,time
2  0,289.365998,0.000300
3  1,204.436001,0.000515
4  2,197.311005,0.000719
5  3,195.286004,0.000920
6  4,194.168999,0.001121
```

n°ope µs s

Démo

Figure: Demo

Résultats

	Création de comptes		Débit et Crédit	
	Latence (µs)	Débit (ope/ms)	Latence (µs)	Débit (ope/ms)
gRPC	85	11	180	6
gRPC + SGX	108	9	283	4

Figure: Résultats comparés

Analyse

- Attendu: SGX légérement + lent.
- Différence de temps faible car c'est le réseau qui domine.

Suite du projet

Propriétées ACID, replay

- Atomicité Cohérence Isolation Durabilité
- Lock (A,C,I ✔)
- problème du replay (D ⊘)

Implémentation gRPC + SGX autogénérée par le compilateur Privagic

- "Colorier" le code
- Comparer le nombre de lignes de code et les performances avec l'implémentation "faîte main"

Compétences acquises

Technologies abordées

- gRPC
- SGX

Structure de données manipulées

- Makefile
- Hashmap
- Dataframe et csv

Management

- Travail en binôme
- Suivi avec les encadrants
- Livrables pour un projet scientifiques

Fin

Merci pour votre attention !