SCBX Cryptography Use Cases: Credit Score Averaging with Differential Privacy and FHE

Daniel Aronoff

MIT

November 19, 2024

Background and Goals

Background

- August - September 2024: SCBX and consultant explored potential use cases to apply cryptography to improve functioning of a bank related activity
- September-October 2024: SCBX and Consultant narrowed the use case to the averaging of credit scores held by two independent entities
- Consultant presented several cryptographic implementations, each with a different profile of attainable privacy and computational complexity
- Two implementation were selected for further exploration

  ↪ A number from each of 2 independent entities is averaged and the solution is provided to a designated recipient. Ideally, the inputs are only known to the agents. The solution is only known to the recipient and all parties can verify that the correct computation was made

Goals

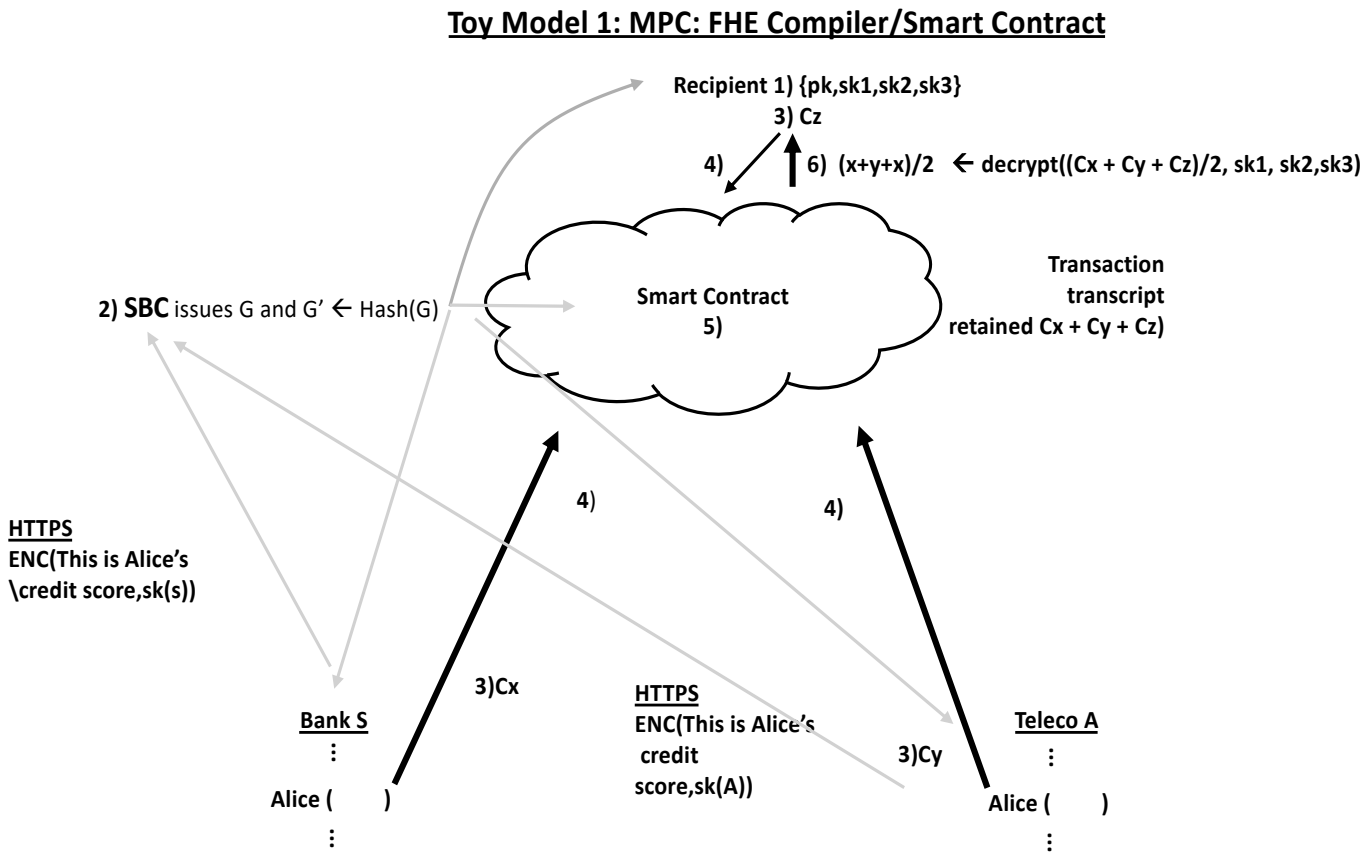
- Code each implementation and run on simulated data (the "toy models")
- Find a business process inside SCB that could benefit from further development of one of the toy models
- **Future work** Design and code a working model to handle the empirical use case

Model Primitives

- HTTPS (internet) is secure communication channel btw the parties over with Enc() data and messages are sent
- Agents: Bank S, Teleco A; SCBX; Inference Engine ("IE") which can be an SCBX computer; Receiver, which can be a third party and any combination of the other agents
- Bank S and Teleco A each message IE and Recipient that it is Alice's data, or ciphertext on Alice's data, that was sent. One (nonexclusive) option is to use symmetric key encryption for this (sender encodes with secret key and receiver decodes with secret key)
- SCB Bank sends evaluation function G to all participants, or hash of G (depending on the case)
- In the toy implementation G $=(\frac{1}{2})$[Teleco A Alice score + Bank S Alice score]

✓ FHE implementations generally have an order of magnitude higher computational complexity than other cryptographic methods e.g. elliptic curve and prime number factoring

Toy Model 1: MPC: FHE Compiler/Smart Contract



Credit Score Model
Cryptographic Goals

- Alice data is private
- IE model parameters are private
- IC can prove integrity of model
  1) Same model used in every credit score computation
  2) Model corresponds to training data

Trust Assumptions

- Alice data can be identified and is accurate
- -- Secure communication channel
- Acceptable information leakage
  - > Depends on setup
  - > Tradeoff btw noise and solution recover

Toy Model 1: MPC: FHE Compiler/Smart Contract

We will select a commercial product that encrypts plaintext into FHE and enables computations to be executed in smart contract designed by the user

The contract can be run on a server or a Turing complete blockchain such as Ethereum

Two platforms to choose from

- Sunscreen `https://docs.sunscreen.tech/intro/intro.html`
- Zama `https://docs.zama.ai/concrete`

Toy Model 1: MPC: FHE Compiler/Smart Contract

Outline of protocol:

1. Recipient issues $\{pk, sk1, sk2, sk3\}$

   Question: can sk be split between participants, or are they constrained to share the same sk? If the former, it is possible to design the mechanism so that only select parties can decode, Need further investigation

2. SBC issues G and hash of evaluation function G to prove it is the correct G

3. Parties encrypt their inputs:

   c_x = FHE.Encrypt(FHE.pk, x) x = Bank A Alice score
   c_y = FHE.Encrypt(FHE.pk, y) y = Teleco S Alice score
   c_z = FHE.Encrypt(FHE.pk, z) z = Recipient score = 0

4. c_x, c_y and c_z are sent to the smart contract and executed

   If the smart contract operates on a public blockchain, the computation is carried out by miners and the participants can observe if the correct function G' is applied

5. Smart contract computes

   (FHE.pk, x + FHE.pk, y + FHE.pk, z)/2

6. The smart contract output is decrypted by secret key $sk = \{sk1, sk2, sk3\}$, which only the Receiver has

Toy Model 1: MPC: FHE Compiler/Smart Contract

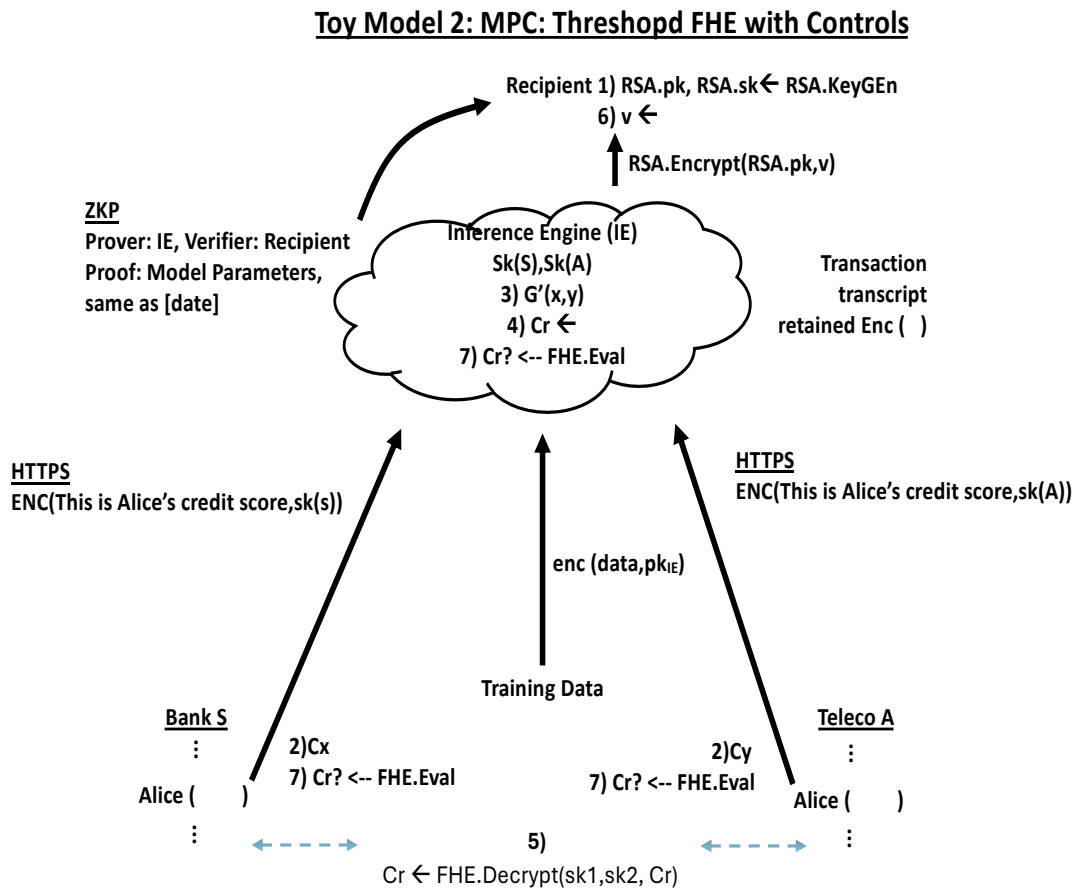
Advantages

- Post quantum secure
- Assurance that smart contract uses the correct function G (or G'), where G is the credit score averaging function
- Off-the-shelf application. Minimal coding required
- Only Receiver sees the solution (iff Receiver can issue multiple secret keys)

Limitations

- Bank S and Teleco A learn the function G
- If smart contract is run on public blockchain anyone can observe function G'(and learn G)

Toy Model 2: MPC: Threshold FHE with controls



Credit Score Model
Cryptographic Goals

- Alice data is private
- IE model parameters are private
- IC can prove integrity of model
  1) Same model used in every credit score computation
  2) Model corresponds to training data

Trust Assumptions

- Alice data can be identified and is accurate
- -- Secure communication channel
- Acceptable information leakage
  - > Depends on setup
  - > Tradeoff btw noise and solution recover

IE Model Training

- ZKP to verify data to train/test

Toy Model 2: MPC: Threshold FHE with controls

Outline of Protocol:

1. Receiver first does (RSA.pk, RSA.sk) $\leftarrow$ RSA.KeyGen and gives RSA.pk to the evaluator
2. Parties encrypt their inputs:

   c_x = FHE.Encrypt(FHE.pk, x) *Bank A Alice score*
   c_y = FHE.Encrypt(FHE.pk, y) *Teleco S Alice score*
3. Evaluator (IE) has in mind some function

   G'(x,y) = A part: v = G(x, y); // u = x+y; v = u * (1/2);
   B part: c_r = RSA.Encrypt(RSA.pk, v);
   return c_r;

   (G' is an encryption of v, i.e., the result of G)

Toy Model 2: MPC: Threshold FHE with controls

4. Evaluator:

   1) computes c_{half} =FHE.Encrypt(FHE.pk, 1/2)

   2a) evaluates encryption of v as:
   c_u = FHE.Add(c_x, c_y)
   c_v = FHE.Mul(c_v, c_{half})

   2b) RSA.Encrypt(RSA.pk, ...) is some circuit (that hard-codes RSA.pk) uses multiplication and addition to produce an encryption of v

   Evaluator uses FHE.Mul and FHE.Add to homomorphically evaluate this circuit on c_v.

   The result of the RSA encryption circuit is some c_r. (see above)

5. Parties then use sk_1 and sk_2 to decrypt c_r

   Result of FHE.Decrypt(sk_1, sk_2, c_r) is r

6. As per above this c_r is RSA.Encrypt(RSA.pk, v) = RSA.Encrypt(RSA.pk, (x+y)/2))

   Receiver decodes and extracts v

7. To ensure that correct G' is used all parties can compute FHE.Eval themselves

   If they don't get to the same c_r, then they abort instead of continuing with the decryption

Toy Model 2: MPC: Threshold FHE with controls

Advantages

- Post quantum secure
- Assurance that evaluator uses correct function G'
- Receiver is only party that learns Alice's credit score
- Not required that all parties stay online through entire process

Limitations

- Complex circuits and computations
- Bank S and Teleco A learn the function G' (and thereby G)
- It may be possible for evaluator to hide the function. This needs further investigation. if so, additional controls may be required to remedy