Fully Homomorphic Encryption

[Group Assignment - 4]

CS6/745: Modern Cryptography

Submitted to: Dr. Yuliang Zheng

Submitted by:

1. Joyanta Mondal (jmondal)
2. Shouzab Khan (Skhan6)
3. Pooja Srinivas (psriniva)
4. Rajendra Mohan (rnavulur)

April 14, 2024

Background about FHE:

Fully Homomorphic Encryption (FHE) is a new type of encryption that allows computations to be carried out directly on encrypted data. This is a major advantage over traditional encryption methods (like symmetric and asymmetric encryption) which require data to be decrypted before you can perform any operations on it. FHE works by performing calculations on the encrypted data itself, generating results that remain encrypted. When decrypted, these results match what you would get if you had performed the same operations on the original, unencrypted data. FHE schemes make use of complex mathematical concepts for encryption, ensuring the security and privacy of the underlying data. The concept of FHE was first introduced by Craig Gentry in 2009, and significant strides have been made towards developing practical FHE systems.

Potential Use Cases:

Fully Homomorphic Encryption has a wide range of possible uses, providing solutions to ongoing issues with data security, secrecy, and privacy. Among the important use cases are:

1. **Secure Cloud Computing:**
   FHE lets you safely store and process sensitive information on the cloud without having to share the actual data. This is a game-changer for sectors like healthcare and finance where privacy is paramount.
2. **Data Analysis with Confidentiality:**
   Using FHE, organizations can analyze huge amounts of data without seeing the individual pieces. This means companies can collaborate and share information while still protecting customer or patient privacy.
3. **Secure Outsourcing of Computation:**
   FHE makes it possible to assign computationally demanding jobs to others while protecting the integrity of the computation results and the privacy of the input data. In situations like data mining, machine learning, and statistical analysis, this is advantageous.
4. **Secure Multiparty Computation:**
   FHE allows groups of people to find insights from their combined data without everyone needing to reveal their own private information. This is known as secure multiparty computation (MPC). This protects privacy while enabling cooperative decision-making and data analysis.

Limitations:

Fully Homomorphic Encryption has a number of limitations that affect its scalability and usefulness, even with its potential applications:

1. **Computationally Costly:**
   Each operation on encrypted data results in additional computational charges, which makes FHE much more computationally costly than typical encryption techniques. For applications that require delay or in real time, FHE may not be feasible due to this overhead.

2. **Key Size and Complexity:**
   Key management and implementation are difficult with FHE methods as they frequently call for big encryption keys and complex mathematical calculations. With limited resources, the overhead related to key generation, storage, and maintenance may make it more difficult to implement FHE.
3. **Performance Constraints:**
   FHE implementations could face issues with memory consumption, processing speed, and scalability in particular. The applicability of FHE to certain use cases and computer environments may be limited by these restrictions. Also, when we run this on a Mac M1 based device, it is not able to execute properly due to compatibility, which may get patched in near future.

```
parallels@ubuntu-linux-22-04-02-desktop:~/fully-homomorphic-encryption$ sudo bazel run //transp
iler/examples/hangman:hangman_client
ERROR: /root/.cache/bazel/_bazel_root/8d6a6530ad751d78683fe2ad497d0a44/external/llvm_toolchain/
BUILD.bazel:65:20: @llvm_toolchain//:local-aarch64-linux: no such attribute 'archive_flags' in
'cc_toolchain_config' rule
ERROR: Analysis of target '//transpiler/examples/hangman:hangman_client' failed; build aborted:
 error loading package '@llvm_toolchain//': Package '' contains errors
INFO: Elapsed time: 0.950s
INFO: 0 processes.
FAILED: Build did NOT complete successfully (0 packages loaded, 0 targets configured)
FAILED: Build did NOT complete successfully (0 packages loaded, 0 targets configured)
    currently loading: @rules_foreign_cc_framework_toolchain_windows_commands//
    Fetching @local_jdk; fetching
```

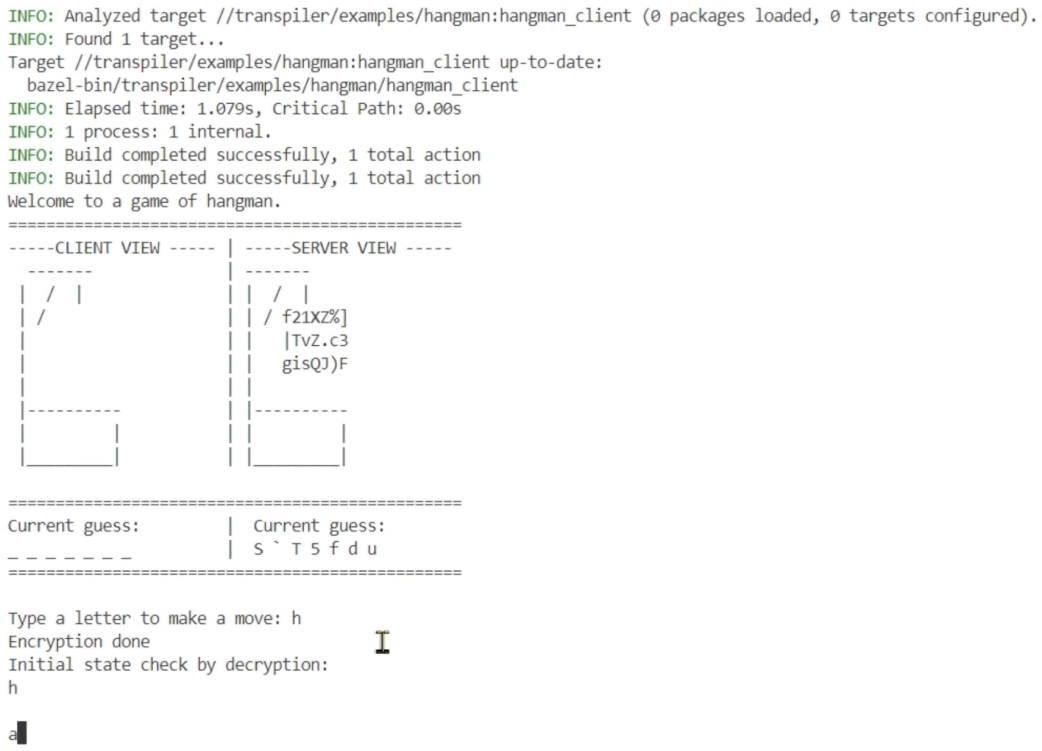
Lessons learned during the research:

Several important lessons were discovered when researching fully homomorphic encryption. These include:

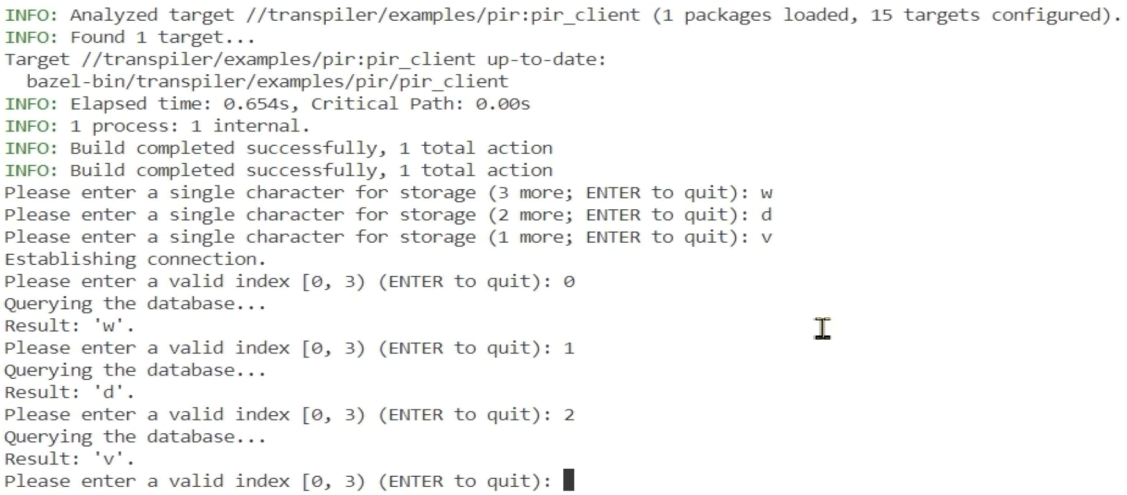
1. **Complexity vs. Practicality:**
   Although FHE provides high security guarantees and permits complex cryptographic features, its processing overhead and complexity present significant deployment challenges in the real world. When analyzing the practicality of FHE solutions, it is crucial to find a balance between practical concerns and security needs.
2. **Performance Choices:**
   When determining whether FHE is appropriate for a certain application, it is important to understand the limitations that exist between security, performance, and usability. The selection of factors, hardware acceleration strategies, and algorithmic decisions all need to be carefully considered when improving FHE implementations for performance.
3. **Complexity of Implementation:**
   FHE schemes rely on advanced mathematical concepts like lattice-based cryptography. This creates a barrier to entry, requiring specialized knowledge to implement and use FHE correctly. Efforts are focused on making FHE more accessible for developers.

Commands:

- Hangman:
  bazel run //transpiler/examples/hangman:hangman_client



- Private information retrieval
  bazel run //transpiler/examples/pir:pir_client



- Reverse string:
  bazel run //transpiler/examples/string_reverse:string_reverse_tfhe_testbench "hello"