Directed Research Statement

Tanmay Ghai

July 27, 2020

1 Introduction

In the world today, especially with the emergence of *Big Data*, the dispersion of data and information into various distributed, multi-user scenarios is pervasive. I believe data is the fuel for the burgeoning *digital economy* and to scale and achieve efficiency on tasks with such magnitudes, businesses, industries, and companies today, rely on many techniques to achieve distributed computation and record linkage. However, with the benefits of shared-distributed computation, one main concern that continues to remain is that of *privacy*. This concern, in fact, impedes many entities from trusting and/or participating in such multi-party/user computations as they fear the data-for-linkage trade-off is a losing one. In specific, one approach in the world of computer science research that seeks to solve this problem is the notion of **Privacy-Preserving Entity Resolution**. It's key ingredient to tackling this issue is that it aims to, without revealing any sensitive information, data, or input, do some computation and entity resolution. Increasingly, the need for such a solution with applications to the real world is essential and of utmost importance. In the words of security expert, Bruce Schneier,

"If you encrypt data in a fully homomorphic cryptosystem, you can ship that encrypted data to an untrusted person and that person can perform arbitrary computations on that data without being able to decrypt the data itself. Imagine what that would mean for cloud computing, or any outsourcing infrastructure: you no longer have to trust the outsourcer with the data!"

2 Definitions

To give clarity and purpose to my directed research statement here, I will provide the following definitions to some terms and key ideas that are central

Entity Resolution: or (ER) is the task of linking, joining, and disambiguating records that correspond to the same real world-entities across and within data sources [Whang, Garcia-Molina].

ER typically has some properties that need to be addressed, and while the set of properties I've chosen isn't formal, it will suffice for the basis of our discussion in this statement:

- *Scalability*: since in today's world, we are customarily dealing with Big Data, it's important we consider and analyze the complexity and performance of our entity resolution scheme. For example, if we were to use PPRL (Privacy-Preserving Record Linkage), our complexity can, in the worst of cases, scale to the range of $|s_1|$ * $|s_2|$ ... * $|s_n|$ for all data sources $s_1 ... s_n$, which can potentially be a huge bottleneck. Clearly, it is vital to keep track of how certain entity resolution schemes scale up with high volumes of data.

- *Linkage Quality*: with high volumes of different, distributed data, another property that can be naturally derived is that of quality. When joining, matching, and linking records across data sets, we need to ensure our scheme for classifying matches and non-matches is accurate and effective.

- *Privacy*: a challenge that arises in solving many ER problems is that of maintaining confidentiality. In all steps of solving ER, privacy needs to be considered and preserved. [Vatsalan, Sehili, Christen, Rahm]

Plaintext & Ciphertext: A *plaintext* is usually a message or some form of "plain data", such as a binary file or an arbitrary-length string encoded as bits, that can be read without a key or any other decryption device; whereas a *ciphertext* is an encrypted version of the plaintext, usually done through an algorithm known as a *cipher*.

Public-key Encryption Scheme: or (PKE) is an encryption scheme that utilizes a pair of public and *private* keys; public which are disseminated widely amongst users and private which are only known to owners. In such a scheme, anyone can encrypt a message utilizing just the public key, but decryption can only happen with the receiver's private key.

Homomorphic Encryption: Using the above definitions, we can formally define homomorphic encryption as follows: given $p_1 ... p_n$ plaintexts which are encrypted into ciphertexts $c_1 ... c_n$ using PKE, homomorphic encryption allows anyone with a public key to be able to compute a function, let's say f, that outputs a new ciphertext encrypting $f(c_1 ... c_n)$, such that they never need to decrypt the data or even know the decryption key. [Harper]

Aside: The most commonly used example for use of homomorphic encryption in a real-world use case is when a user wants to send data to the cloud for some computation/processing, but is not trusting of a service provider. Using a homomorphic scheme as described above, the data owner can encrypt their data, and send it to the server where the computations can be done without ever decrypting the data itself. The service provider sends the encrypted results back to the owner who is able to decrypt the results with their own private key. The most efficient homomorphic schemes known today are the Brakersi-Gentry-Vaikuntanathan (BGV) Brakersi/Fan-Vercauteren, (BFV) and, Cheon-Kim-Kim-Song (CKKS). I will refer to these schemes later in part 3.

Fully Homomorphic Encryption Scheme: or (FHE); While the definition above is useful, it is not strong enough to ensure our scheme is in fact *fully homomorphic*. The above definition ensures, for trivial examples, that we will output an evaluated function f on top of the encrypted data. However, to ensure more stringently that this given output indeed reveals nothing about the given function, we need to require **circuit privacy**. Circuit Privacy, specifically, refers to the fact that the evaluated ciphertext in FHE should also hide the function f in addition to what is leaked through the computation of $f(c_1 ... c_n)$. [Bourse, Pino, Minelli, Wee] There are many ways of doing this that I will hope to explore during my DR, but one example of a methodology to do this is known as *garbled circuits*. [Gentry]

To further establish how entity resolution, privacy, and FHE schemes fit in together, I will go through an entity resolution problem to show how we might use FHE to address the privacy concerns that may arise. Consider the following situation:

Say there is a medical researcher who is trying to investigate how health care costs in America disproportionately affect people of certain races and income levels. Clearly, to solve this problem, data would be required from multiple different data sources: medical, financial institutions, and perhaps even Census/demo-graphical data. Amongst the data that this researcher would require are some various personal details (such as name, age, DOB, SSN, address, etc); these are commonly known as QID's (quasi-identifiers) or data fields that would be common across data sources and would be easy to link over. The main challenge (privacy wise), however, is that law enforcement as well as the various entities providing the data to the researcher do not want any of this personal information to be disseminated to any third party (see HIPAA and GDPR standards). Thus, the researcher has to figure out how to perform this record linkage without revealing any sensitive information to an organization, party, or adversary in the linkage process. Many methods to approach this problem exist today, including data pre-processing/filtering, masking of data, blocking, and bloom filtering which is a data filtering process through a bit vector which stores values into a data structure mapped through a set of hash functions. While there are some privacy assurances with these methods, they still remain susceptible to various adversarial attacks (such as *frequency attacks* where the frequency distribution of masked values is matched against unmasked values to infer the original values themselves. In particular, Bloom filters are probably the most widely accepted solution (practically speaking) today, but are unfortunately not provably safe [Kroll, Steinmetzer]. Here's where FHE can come in, to theoretically, achieve the goal of a *fully private* record linkage.

To formalize this problem, let's say we are dealing with three databases F for the financial institution, M for medical, and D for demographical information and our researcher can be represented as client c. Our goal is to allow for the client to request and make queries to all three databases such that the databases should learn nothing about the query and the client should learn nothing else about the contents of the database. In this scenario, the client c along with all parties F, M, and D would all need to jointly agree on an FHE scheme where c would generate a public-private key pair. A protocol using FHE to retrieve pertinent information to the client c can be described below.

1. c requests ith position of database d_i where d_i iterates between F, M, and D.
2. Let $p = p_1 ... p_n$ represent plaintexts in our fully homomorphic cryptosystem. Then, c can request queries of the form $Q = q_i$ where i goes from 1 to n; each q_i representing a ciphertext that is sent over to the database d_i.
3. The database will respond with
   $R = \sum_{n=1}^{n} x_i * q_i$ where $X = x_1 ... x_n$ can represent the ith bit represented in the database.
   Finally, the client c can recover the desired bit via decryption.
   $D(R) = D(\sum_{n=1}^{n} x_i * q_i)$.
   This protocol is easily seen to be private as the only information given to each database d_i is a list of ciphertexts from indistinguishable distributions. Similarly, the client c only receives information about the ith bit requested.[Ostrovsky and Skeith III]

This specific example is known as *Secure Private Information Retrieval* and has applications that extend towards another related problem: *Privacy-Preserving Record Linkage* (PPRL). Both have tremendous research around them such as this and this, respectively.

3 Research Proposal

While this may not be the only thing I explore in my DR, for the purposes of this proposal, I will focus my thoughts around a specific, yet simple problem using fully homomorphic encryption. Under a set of computation tools, what's known as *secure multi party computation* (SMPC), a technique known as **Private Set Intersection** exists. PSI enables two or more separate entities, each with a set of data, to *securely* compute the intersection of their data without giving up any form of individual privacy.

An example of PSI can be set up as follows; for simplicity, we will establish a two party PSI example with one *sender* and one *receiver*. The sender has a set of items $X = x_1...x_n$ of size, let's say N and the receiver has a set of items $Y = y_1...y_m$ of size M. The goal is to compute the intersection, output nothing to the sender and $X \cap Y$ to the receiver. A more formal definition of such a protocol is detailed below:
Input: Receiver with input Y, Sender with input X, both consist of items represented as δ-length bit strings
Output: Receiver outputs $X \cap Y$, sender outputs null

1. Sender and receiver agree on an FHE scheme. As established in PKE, the receiver will generate a public-private key pair and keep the private key secret.

2. The receiver will encrypt each element y_i in its set using the FHE. It will then send $c_1...c_m$ ciphertexts generated from the scheme to the sender.

3. The sender will compute the intersection; for each incoming c_i
   - samples a plaintext element p_i
   - homomorphically computes ciphertexts as such
   $d_i = r_i \prod(c_i - x)$ for all $x_i \in X$.

4. These ciphertexts $d_1...d_n$ are returned to the receiver, who decrypts them and outputs $X \cap Y$.

This is a generalization of the *Basic PSI protocol* established in this paper. I will seek to implement some algorithms as such utilizing the PALISADE cryptographic library. Over the course of my work, I may also investigate/utilize these other two libraries: Conclave, a query compiler that optimizes SMPC queries by transforming them into small data-parallel and local cleartext processing steps and Microsoft SEAL, a composed set of encryption libraries allowing direct computations to be performed on encrypted data.

In addition to the three libraries mentioned above, FHE implementations have been developed in various other capacities. Some examples are detailed below and an up to date list can be found here.

- PALISADE: supports the BFV, BGV, CKKS schemes as well as boolean circuit evaluation techniques and optimizations.
- Microsoft SEAL: implements and supports the BFV and CKKS schemes.

- HElib: implements both the BGV and CKKS schemes as well as Smart-Vercauteren ciphertext packing techniques.
- HEAAN: implements the CKKS scheme.
- Lattigo: implements BFV and CKKS in Go.
- $\Lambda.\lambda$: Haskell cryptography library that supports FHE implementations and lattice based cryptography, generally.

Note: FHE cryptosytems can largely be categorized into those that are based on lattice-based cryptography and those that aren't. The BGV, BFV, and CKKS schemes are all lattice-based cryptographic schemes, which means they are linear, can perform efficient parallel operations, and are quantum-safe as of today (can resist quantum attacks).

4 Conclusion

Fully Homomorphic Encryption is certainly a powerful tool and has recently emerged as an area in computer science research that can broadly apply to some cutting edge, real-world use cases, with Entity Resolution only being one of them. Some examples to be noted: *Private Contact Discover, DNA testing/pattern matching, and even contact tracing for COVID-19.* [OpenMined] Though, fully homomorphic schemes are not widely used in practice or industry anywhere today, I hope my deep dive into this fascinating world at the intersection of cryptography, cyber-security, and distributed systems can help progress this field a little further.

References

[Whang, Garcia-Molina] *Joint Entity Resolution*, Google Research & Stanford University

[Vatsalan, Sehili, Christen, Rahm] *Privacy-Preserving Record Linkage for Big Data: Current Approaches and Research Challenges*, Australian National University & University of Leipzig

[Harper] *Fully Homomorphic Encryption Scheme*, University of Washington

[Bourse, Pino, Minelli, Wee] *FHE Circuit Privacy Almost For Free*, ENS, CNRS, INRIA, and PSL Research University, Paris, France

[Gentry] *A Fully Homomorphic Encryption Scheme*, Stanford University

[Kroll, Steinmetzer] *Automated Cryptanalysis of Bloom Filter Encryptions of Health Records*, Research Methodology Group, University of Duisburg-Essen, Essen, Germany

[Ostrovsky and Skeith III] *A Survey of Single-Database PIR: Techniques and Applications*, UCLA Coomputer Science

[OpenMined] *A Privacy-Preserving Way to find the Intersection of two DataSets*, OpenMined Privacy-Preserving Data Science