Fundamentals of Cryptography **Fall 2018**

Project 10: Homomorphic Encryption

Lecturer: Prof. Wenfei Wu *Scribes: Zhizhou Ren, Xiaoyao Zou*

1 Introduction

Recent advances in machine learning requires the extensive use of private and personally identifiable information, especially in domain of biomedical and genetic research. There are two common application scenarios: 1) someone who owns lots of private data, and wants to adopt a cloud computation server. 2) or someone wants to adopt a pre-trained inference model, and apply on his private data.

One way to ensure trust in the scenarios above is through storage and disclosure of only secure, encrypted data. From a data science perspective, the problem with employing cryptographic methods to improve trust is that the data must at some point be decrypted for use in a further analysis. However, recent cryptography research in the areas of homomorphic encryption are showing exciting potential to bypass this. An encryption scheme is said to be homomorphic if certain mathematical operations can be applied directly to the cipher text in such a way that decrypting the result renders the same answer as applying the function to the original plaintext data.

1.1 Definition

To simplify the notation, only public key schemes are considered. An encryption scheme is said to be homomorphic for some operation $\otimes \in \mathcal{F}_M$, if there exists a corresponding operation $\oplus \in \mathcal{F}_C$ acting in cipher text space satisfying the property

$$\forall m_1, m_2 \in M,\ D(k_s, E(k_p, m_1) \oplus E(k_p, m_2)) = m_1 \otimes m_2$$

where $E(k_p, m)$ and $D(k_s, c)$ are encoding and decoding function, k_p and k_s are public and secret key.

Recall a well-known result from computational complexity theory, in space $M = \{0, 1\}$, addition and multiplication are corresponding to XOR and AND, which is sufficient to construct arbitrary binary circuits, and have universal computation ability. Thus a common setting is to implement $\mathcal{F}_M = \{+, \times\}$, and encryption method are distinguished by

- Partially homomorphic encryption (PHE): either addition or multiplication is implemented.
- Fully homomorphic encryption (FHE): both addition and multiplication are implemented.

1.2 Security

The security issues in homomorphic cryptanalysis are always considered in these two settings:

- Semantical Secure formalized by Shannon (1949) provides a systematic method to characterize the secure property for which the knowledge of a ciphertext does not give any information either about the corresponding plaintext or about the key.

- Diffie & Hellman (1976) proposed a method to distinguish cryptosystem according to the capacity of an attacker, and Chosen-Plaintext Attack (CPA) corresponds to attacker can choose an unlimited number of plaintexts and obtain corresponding ciphertexts.

2 Practical Difficulty

2.1 Depth of Operations

A counterintuitive fact against the definition is that, in general sense,

$$E(k_p, m_1) \oplus E(k_p, m_2) \neq E(k_p, m_1 \otimes m_2)$$

An intuitive explanation follows: most of encryption methods add noise in ciphertext due to semantic secure (recall the padded RSA discussed in the course), and computation will increase these noise, the message may be hard to decrypted after a long series of operations.

Many FHE methods only support limited number of operations (e.g. unlimited additions and only one multiplication), and this property is the greatest contribution of the first complete FHE method proposed by Gentry & Boneh (2009).

2.2 Message Space

A negative fact is that, there are currently no schemes which will directly encrypt arbitrary values in $\mathbb{R}$. Boolean circuit is computationally universal in complexity theory, but expensive in practice.

However, there are schemes which have an expanded some specific message space such as $\mathbb{Z}/n\mathbb{Z}$ or other rings. A method which can be used to increase the size of the message space is via the Chinese Remainder Theorem (CRT) as a means of representing a large integer. More formally, it follows $\mathbb{Z}/M \cong \mathbb{Z}/m_1 \times \cdots \times \mathbb{Z}/m_k$, and another advantage of CRT is that parallel can be directly derived by its property.

2.3 Ciphertext Size

There will be a substantial inflation in the size of the message after encryption, often by several orders of magnitude. As a concrete example, the usual representation of an integer in a computer requires only 4 bytes of memory, but such a message is encrypted under the scheme given by Fan & Vercauteren (2012) which is one of the most practical FHE method, the result in cipher texts occupying 65536 bytes. Consequently, a 1MB data set will occupy nearly 16.4GB encrypted.

2.4 Computational Cost

Most of homomorphic encryption method use a much larger ciphertext space than message space, such as polynomial rings. Consequently, arithmetic operations are substantially more costly than standard arithmetic.

3 Milestones

3.1 Partially Homomorphic Encryption

Two classical encryption discussed in the course can directly provide homomorphic multiplication.

- Textbook RSA (Rivest et al., 1978) supports homomorphic multiplication but not secure.
- EIGamal (ElGamal, 1985) supports multiplication and satisfies semantic secure and CPA.

Although the addition and multiplication can be converted by primitive root in a modular field, but discrete logarithm is a hard problem. The following cryptosystems are the state of art method to implement homomorphic addition, and both of them satisfy semantic secure and CPA.

- Goldwasser-Micali cryptosystem (Goldwasser & Micali, 1982)
  1. Find two large primes p, q, and $n = pq$.
  2. Select a quadratic nonresidue g, characterized by Jacobi symbol (Jacobi, 1846).
     $$\left(\frac{g}{n}\right) = \pm 1 \equiv g^{\frac{(p-1)(q-1)}{4}} \ mod\ n$$
     similar with Legendre symbol, $\left(\frac{g}{n}\right) \equiv 1$ means g is quadratic residue.
  3. Key Generation: $k_p = (n, g)$, $k_s = (p, q)$.
  4. Encryption: for $m \in \{0, 1\}$, $E(k_p, m) = g^m r^2 \ mod\ n$, where $r \in \mathbb{Z}_n^*$ is a random integer.
  5. Decryption: determine whether c is quadratic residue by Jacobi symbol, $m = [\left(\frac{c}{n}\right) = -1]$.
  6. Addition: $m_1 + m_2 \equiv D(k_s, E(k_p, m_1) \times E(k_p, m_2)) \ mod\ 2$.

  Proof. (correctness)
  Whether $c \equiv g^m r^2 \ mod\ n$ is quadratic residue only depends on g^m, and g is a quadratic nonresidue number we prepared, so $m = 1$ if and only if $\left(\frac{c}{n}\right) = -1$, otherwise $m = 0$.
  And for addition operation, $D(k_s, E(k_p, m_1) \times E(k_p, m_2)) = D(k_s, g^{m_1+m_2} r_1^2 r_2^2) = m_1 + m_2$. □

- Parillier Cryptosystem (Paillier, 1999)
  1. Find two large primes p, q, and $n = pq$, $\lambda = lcm(p-1, q-1)$.
  2. Select a random number g, satisfying $n | ord_{n^2}(g)$.
  3. Key Generation: $k_p = (n, g)$, $k_s = \lambda$.
  4. Encryption: for $m \in \mathbb{Z}_n$, $E(k_p, m) = g^m r^n \mod n^2$, where $r \in \mathbb{Z}_n^*$ is a random integer.
  5. Decryption:
     $$m = D(p_s, c) = \frac{L(c^\lambda \ mod\ n^2)}{L(g^\lambda \ mod\ n^2)} \ mod\ n$$
     where $L(x) = \frac{x-1}{n}$.
  6. Addition: $m_1 + m_2 \equiv D(k_s, E(k_p, m_1) \times E(k_p, m_2)) \ mod\ n$.

Proof. (correctness)
Recall that $\lambda = lcm(p-1, q-1)$ is the Carmichael function of n, it can be substituted by Euler function $\varphi(n) = (p-1)(q-1)$ to simplify its notation.
And in this setting $\forall g \in \mathbb{Z}_n^*$,

$$g^\lambda \equiv 1 \ mod \ n \equiv 1 + un \ mod \ n^2 \qquad g^{\lambda n} \equiv 1 \ mod \ n^2$$

where $u \in \mathbb{Z}_n$ is an integer corresponding to g, and $u \neq 0$ due to $n | ord_{n^2}(g)$.
Consider

$$c^\lambda = g^{\lambda m} r^{\lambda n} \equiv (1 + un)^m \ mod \ n^2 \equiv 1 + unm \ mod \ n^2$$

Thus $L(c^\lambda \ mod \ n^2) = um$, $L(g^\lambda \ mod \ n^2) = u$, $D(p_s, c) = m$.
And for addition operation, $D(k_s, E(k_p, m_1) \times E(k_p, m_2)) = D(k_s, g^{m_1+m_2} r_1^n r_2^n) = m_1 + m_2$. □

3.2 Fully Homomorphic Encryption

As mentioned before, a FHE scheme enables arbitrary computation to be performed on encrypted data. FHE have been regarded as the holy grail of modern cryptography for a long time. There is no complete FHE scheme was construable until the breakthrough work of Gentry & Boneh (2009).

- As an important result, Boneh et al. (2005) constructs a scheme allowing unlimited additions and a single multiplication. It's the first scheme which supports both operations.
- The breakthrough work given by Gentry & Boneh (2009) activated this three decade old problem, they proposed an encryption method based on ideal lattices and their blueprint contains the following two stages:
    1. Firstly to construct a somewhat homomorphic encryption (SWHE) scheme which can homomorphically evaluate low-degree polynomials. More specifically, SWHE schemes allow only a limited number of homomorphic computations on ciphertexts (e.g. many additions and a small number of multiplications).
    2. Recall that in almost proposed FHE schemes, encryption is a process of adding noise to plaintexts, and this noise would increase in the process of any homomorphic computation and when it becomes too large, correct decryption is impossible even with the right decryption key. So the other important stage is to reduce noise in ciphertext, this step called bootstrapping in their work, and it's very costly in computation.

  Although this scheme has solved FHE in a theoretical perspective, but it's fully intractable in practical scenarios due to its despairing computation cost. Based on this work, Stehlé & Steinfeld (2010) improved the analysis on Sparse Subset Sum Assumption against lattice attacks more aggressively and introduced a probabilistic decryption algorithm with low multiplicative degree, thus produced a faster FHE scheme.
- Instead of ideal lattices, Van Dijk et al. (2010) proposed a FHE scheme over the integers. FHE over the integers has the advantage of conceptual simplicity. However, this scheme is also far from practical since it involves large public key size and suffers low efficiency.
- Most of current FHE implementation libraries are profit from the contribution of Fan & Vercauteren (2012), they improved the method given by Brakerski (2012) which is based on learning with errors (LWE) problem, and proposed a much practical scheme based on Decision Ring learning with errors (Decision-RLWE) problem (Regev, 2009), which is connected to classical cryptography hardness results. The results in their paper provide a sound theoretical basis for practical implementations.

4 Implementations

Many homomorphic schemes can be non-trivial to implement and hard to derive real application. Most of public implementations are releases of software which was written for a specific paper, whilst there are a small number of libraries or packages enabling reuse.

- 'libfhe' (Minar, 2010) is a very compact single C file library implementing Gentry (2010).
- 'Scarab' (Shai Halevi, 2014) provides a C++ library implementing Brakerski (2012).
- Lepoint (2014) provides a C++ library for Fan & Vercauteren (2012) and Bos et al. (2013).
- Aslett (2014) provides an easy to use interface to begin developing and testing statistical methods in a homomorphic environment by R package.

5 Encrypted Statistics

In the recent years, some work has emerged on statistical methods for homomorphically encrypted data. In these methods, most of heavy computation can be done in a cloud server, and a small number of hard operations such as division need to be executed by clients themselves.

- Graepel et al. (2012) implements an encrypted version of Linear Discriminant Analysis. The algorithm has been rewritten in a way that divisions are avoided, some of the computation is done offline by the client after decrypting results returned by the cloud server.
- Lauter et al. (2014) observed that it is possible to analyse genomic data in a privacy preserving framework and provide some examples of algorithms in statistical genetics which are implementable under the restrictions of homomorphic encryption, including the Cochran-Armitage trend test, the expectation-maximisation algorithm and measures of goodness-of-fit and linkage disequilibrium.
- For prediction tasks, covariates are encrypted and sent to the server where a pre-trained model has been saved, the results then returned to the client for decryption. These applications includes hidden Markov models (Pathak et al., 2011) and logistic regression (Bos et al., 2013)

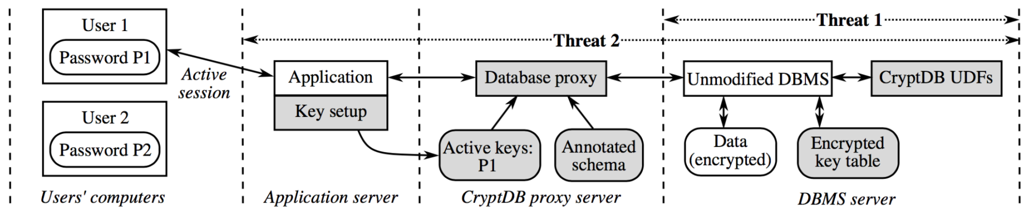
6 CryptDB: An application on Database Management

Theft of private information is a significant problem, particularly for online applications. CryptDB Popa et al. (2011) is a system that explores an intermediate design point to provide confidentiality for applications that use database management systems (DBMSes). CryptDB consists of a DBMS server and a separate application server. The application server runs the application code and issues DBMS queries on behalf of one or more users.

CryptDB uses three key ideas:

- Execute SQL queries over encrypted data
- Adjustable query-based encryption
- Chain encryption keys to user passwords

6.1 Security Overview



CryptDB is designed to defend two types of threats:

The first threat is a curious database administrator who tries to learn private data by snooping on the DBMS server.

The second threat is an adversary that gains complete control of application and DBMS servers. In this case, CryptDB cannot provide any guarantees for users that are logged into the application during an attack, but can still ensure the confidentiality of logged-out users' data.

We will discuss them separately in the next two subsections.

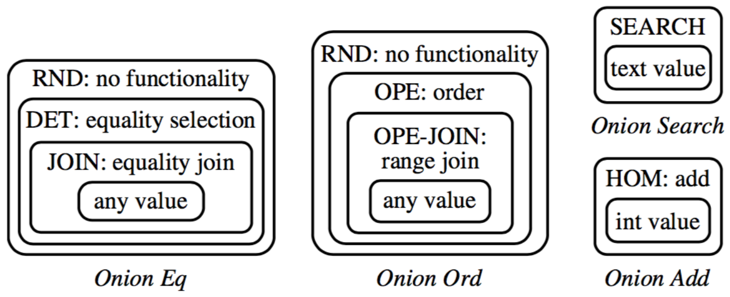
6.2 Queries On Encrypted Data

6.2.1 Basic Encryption Types

- Random (RND): an encryption method that satisfies IND-CPA(indistinguishability under an adaptive chosen-plaintext attack). CryptDB uses AES in CBC mode with a random IV.
- Deterministic (DET): an encryption method that generates the same ciphertext for the same plaintext. This encryption only leaks which encrypted values correspond to the same data value. CryptDB uses a zero IV for AES-CMC.
- Order-preserving encryption (OPE): an encryption method that satisfies If $x < y$, then $OPE_K(x) < OPE_K(y)$. It is used to perform range queries, min/max queries, sort queries, etc. CryptDB uses the scheme in Boldyreva et al. (2009)
- Homomorphic encryption (HOM): an IND-CPA secure encryption method that can perform plus operations between cells. CryptDB uses Paillier cryptosystem Paillier (1999).
- Join (JOIN): an encryption method that allows equality joins between two columns. It is discussed in the "Computing Joins" subsection.
- Word search (SEARCH): For each column needing SEARCH, split the text into keywords. Then remove repetitions, randomly permute the positions of the words and encrypt each of the words using the protocol in Song et al. (2000), padding each word to the same size. It is nearly as secure as RND, with the difference being the leak of the number of keywords.

6.2.2 Adjustable Query-based Encryption

The goal of CryptDB is to use the most secure encryption schemes that enable running the requested queries. So the server shouldn't know more information than needed to perform operations.



The idea is to encrypt each data item in one or more "onions", each layer of each onion enables certain kinds of functionality.

For each layer of each onion, the proxy uses the same key for encrypting values in the same column, and different keys across tables, columns, onions, and onion layers. So the proxy can perform operations on a column without having to compute separate keys for each row that will be manipulated, also the server can't learn any additional information in other columns. All of these keys are derived from the master key.

Each onion starts out encrypted with the most secure encryption scheme and is removed when needed by sending the corresponding onion key to the server.

6.2.3 Executing over Encrypted Data

When we use HOM to handle additions, the values in the OPE and DET onions would become stale. In fact, any hypothetical encryption scheme that simultaneously allows addition and direct comparison on the ciphertext is insecure: the server can repeatedly add one to each field until it becomes equal to some other value in the same column.

There are two approaches:

If a column is incremented and then only projected: When a query requests the value of this field, the proxy should request the HOM ciphertext from the Add onion, instead of ciphertexts from other onions, because the HOM value is up-to-date.

If a column is used in comparisons after it is incremented: Replace the update query with two queries: a `SELECT` of the old values to be updated followed by an `UPDATE` setting the new values. This strategy would work well for updates that affect a small number of rows.

6.2.4 Computing Joins

To perform joins between two encrypted columns, the columns should be encrypted with the same key so that the server can see matching values between the two columns. To solve this problem, we introduce a new cryptographic primitive, JOIN-ADJ, which allows the DBMS server to adjust the key of each column at runtime.

JOIN-ADJ is non-invertible, so we define the JOIN encryption scheme as JOIN(v) = JOIN-ADJ(v)$||DET(v)$. This construction allows the proxy to decrypt a JOIN(v) column to obtain v by decrypting the DET component, and allows the DBMS server to check two JOIN values for equality by comparing the JOIN-ADJ components.

Each column is initially encrypted at the JOIN layer using a different key. When a query requests a join, the proxy gives the DBMS server an onion key to adjust the JOIN-ADJ values in one of the two columns. After the adjustment, the columns share the same JOIN-ADJ key. The DET components of JOIN remain encrypted with different keys.

JOIN-ADJ is transitive: if the user joins columns A and B and then joins columns A and C, the server can join B and C. In order to do this, CryptDB adjusts all the columns in the same set with column A so that all the columns in both sets share the same key after adjustment.

6.2.5 JOIN-ADJ construction

JOIN-ADJ uses elliptic-curve cryptography. JOIN-ADJ$_K(v)$ is computed as

$$\text{JOIN-ADJ}_K(v) := P^{K \cdot PRF_{K_0}(v)}$$

where K is the key, P(public) is a point on an elliptic curve, and PRF_{K_0} is a pseudo-random function with K_0 being a key that is the same for all columns.

When a query joins columns c and c', each having keys K and K' at the join layer, the proxy computes $\Delta K = K/K'$ (in an appropriate group) and sends it to the server. Then, given JOIN-ADJ$_{K'}(v)$ (the JOIN-ADJ values from column c') and ΔK, the DBMS server adjusts the key in c' by computing:

$$\begin{aligned}(\text{JOIN-ADJ}_{K'}(v))^{\Delta K} &= P^{K' \cdot PRF_{K_0}(v) \cdot (K/K')} \\ &= P^{K \cdot PRF_{K_0}(v)} \\ &= \text{JOIN-ADJ}_K(v)\end{aligned}$$

6.3 Multiple Principles

We now extend the threat model to the case when the application and proxy are also untrusted (threat 2). Note that CryptDB cannot provide any guarantees for users that are logged into the application during an attack in this case.

6.3.1 Basic Idea

In CryptDB, each user has a key that gives access to his data. CryptDB encrypts different data items with different keys, and enforces the access control policy using chains of keys starting from user passwords and ending in the encryption keys of SQL data items. The operations to the database can be done in the following steps:

- When a user logs in, he provides his password to the proxy.
- The proxy uses this password to derive onion keys to process queries on encrypted data and to decrypt the results.
- The proxy can now decrypt the data that the user has access to.
- The proxy gives the decrypted data to the application, which can now compute on it.

- When the user logs out, the proxy deletes the user's key.

Each sensitive field is encrypted with the key of the principal. CryptDB encrypts the sensitive field with onions in the same way as for single-principal CryptDB, except that onion keys are derived from a principal key as opposed to a global master key.

6.3.2 Message Passing

Each principal is associated with a secret, randomly chosen key. If principal B speaks for principal A, then principal A's key is encrypted using principal B's key, and stored as a row in the special access keys table in the database. This allows principal B to gain access to principal A's key.

The key of each principal is a combination of a symmetric key and a public private key pair. In the common case, CryptDB uses the symmetric key to encrypt any data and other principals' keys accessible to this principal (which is faster). However, if some principal is not currently online, CryptDB does not have access to user 1's key, so it will not be able to encrypt message m's key with user 1's symmetric key. In this case, CryptDB looks up the public key of the principal in a second table and encrypts message m's key using user 1's public key. When user 1 logs in, he will be able to decrypt the key for message m.

6.4 Discussion

CryptDB supports most relational queries and aggregates on standard data types. Additional operations can be added to CryptDB by extending its existing onions, or adding new onions for specific data types.

There are certain computations CryptDB cannot support on encrypted data. For example, it does not support both computation and comparison on the same column.

In multi-principal mode, CryptDB cannot perform server-side computations on values encrypted for different principals, even if the application has the authority of all principals in question.

7 Conclusion

In this report, we give a review on homomorphic encryption developments and its privacy preserving applications. In summary, although there are many breakthrough progress in the last decade, the practical limitations of homomorphic encryption schemes also impede most of existing techniques to translate into a corresponding secure algorithm. The most important requirement is to provide better compatibility with non-integer data, and accelerate the process of those operations which can not be implement in homomorphic encryption directly, such as division and comparison, reduce the ciphertext space and distributed computation can also be considered to improve the performance.

In conclusion, homomorphic encryption schemes were considered highly valuable in ensuring data privacy since they allow meaningful computations to be performed in an encrypted form without decrypting the data. These method already achieves great success in theoretical perspective, improving its efficiency in real privacy preserving applications may be an important direction and challenge of this field.

References

Louis Aslett. An r package for fully homomorphic encryption. 2014. URL http://www.louisaslett.com/HomomorphicEncryption/.

Alexandra Boldyreva, Nathan Chenette, Younho Lee, and Adam Oneill. Order-preserving symmetric encryption. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 224–241. Springer, 2009.

Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-dnf formulas on ciphertexts. In *Theory of Cryptography Conference*, pp. 325–341. Springer, 2005.

Joppe W Bos, Kristin Lauter, Jake Loftus, and Michael Naehrig. Improved security for a ring-based fully homomorphic encryption scheme. In *IMA International Conference on Cryptography and Coding*, pp. 45–64. Springer, 2013.

Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical gapsvp. In *Advances in cryptology–crypto 2012*, pp. 868–886. Springer, 2012.

Whitfield Diffie and Martin Hellman. New directions in cryptography. *IEEE transactions on Information Theory*, 22(6):644–654, 1976.

Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE transactions on information theory*, 31(4):469–472, 1985.

Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. *IACR Cryptology ePrint Archive*, 2012:144, 2012.

Craig Gentry. Computing arbitrary functions of encrypted data. *Communications of the ACM*, 53(3):97–105, 2010.

Craig Gentry and Dan Boneh. *A fully homomorphic encryption scheme*, volume 20. Stanford University Stanford, 2009.

Shafi Goldwasser and Silvio Micali. Probabilistic encryption &amp; how to play mental poker keeping secret all partial information. In *Proceedings of the fourteenth annual ACM symposium on Theory of computing*, pp. 365–377. ACM, 1982.

Thore Graepel, Kristin Lauter, and Michael Naehrig. Ml confidential: Machine learning on encrypted data. In *International Conference on Information Security and Cryptology*, pp. 1–21. Springer, 2012.

Carl Gustav Jacob Jacobi. Über die kreistheilung und ihre anwendung auf die zahlentheorie. *Journal für die reine und angewandte Mathematik*, 30:166–182, 1846.

Kristin Lauter, Adriana López-Alt, and Michael Naehrig. Private computation on encrypted genomic data. In *International Conference on Cryptology and Information Security in Latin America*, pp. 3–27. Springer, 2014.

Tancrde Lepoint. Proof-of-concept implementation of a homomorphic simon encryption using yashe and fv leveled homomorphic cryptosystems. 2014. URL https://github.com/tlepoint/homomorphic-simon.

Jan Minar. 'libfhe'. 2010. URL https://github.com/rdancer/fhe/tree/master/libfhe.

Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 223–238. Springer, 1999.

Manas A Pathak, Shantanu Rane, Wei Sun, and Bhiksha Raj. Privacy preserving probabilistic inference with hidden markov models. In *ICASSP*, pp. 5868–5871, 2011.

Raluca Ada Popa, Catherine Redfield, Nickolai Zeldovich, and Hari Balakrishnan. Cryptdb: protecting confidentiality with encrypted query processing. In *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*, pp. 85–100. ACM, 2011.

Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM (JACM)*, 56(6):34, 2009.

Ronald L Rivest, Adi Shamir, and Leonard Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

Victor Shoup Shai Halevi. 'helib'. 2014. URL https://github.com/shaih/HElib.

Claude E Shannon. Communication theory of secrecy systems. *Bell system technical journal*, 28(4):656–715, 1949.

Dawn Xiaoding Song, David Wagner, and Adrian Perrig. Practical techniques for searches on encrypted data. In *Security and Privacy, 2000. S&P 2000. Proceedings. 2000 IEEE Symposium on*, pp. 44–55. IEEE, 2000.

Damien Stehlé and Ron Steinfeld. Faster fully homomorphic encryption. In *International Conference on the Theory and Application of Cryptology and Information Security*, pp. 377–394. Springer, 2010.

Marten Van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 24–43. Springer, 2010.