Homomorphic Encryption

KTH Thesis Report

Daniel Sanaee

Authors

Daniel Sanaee <dsanaee@kth.se>
Mathematics
KTH Royal Institute of Technology

Place for Project

Stockholm, Sweden

Examiner and supervisor

Johan Håstad
KTH Royal Institute of Technology

Abstract

Write an abstract. Introduce the subject area for the project and describe the problems that are solved and described in the thesis. Present how the problems have been solved, methods used and present results for the project. Use probably one sentence for each chapter in the final report.

The presentation of the results should be the main part of the abstract. Use about ½ A4-page. English abstract

Keywords

Template, Thesis, Keywords ...

Abstract

Svenskt abstract

Nyckelord

Mall, Examensarbete, Nyckelord ...

Acknowledgements

Write a short acknowledgements. Don't forget to give some credit to the examiner and supervisor.

Acronyms

Contents

Chapter 1

Theory

INTRODUCTION

1.1 Computation theory

In this section, it is assumed that the reader has prior knowledge of deterministic- and probabilistic Turing machines as a model of computation (an excellent introduction can be found in [**Gol01**]). We introduce an alternative model of computation based on sets of circuits for the purpose of protection against stronger adversaries. We include the basics needed to understand cryptography and homomorphic encryption in particular.

Digital logic

Definition 1 (Circuit)**.** For $n, m \in \mathbb{N}$ and any field $(\mathbb{F}, +, \times)$, an arithmetic circuit is a vector-valued polynomial function $C\colon \mathbb{F}^n \to \mathbb{F}^m$.

A circuit C is represented by a finite directed acyclic graph with n source nodes (the n inputs) and m sink nodes (the m outputs). The internal nodes of the circuit are called gates. For more details about the structure of a circuit, see [**goldreich_2008**] or [**MF21**]. The number of nodes in C is called its *size* and is denoted $|C|$. The longest path in C is called its *depth*. A circuit is called *Boolean* when the field is $\mathbb{F}_2$ and each gate takes at most 2 inputs. Boolean circuits and arithmetic circuits are equivalent in the sense that the set of functions that can be computed by an arithmetic circuit is equal to the set of functions that can be computed by a Boolean circuit.[1] If the input is a string

[1] A Boolean circuit is an arithmetic circuit. Conversely, any Boolean circuit can be simulated by

of bits, we assume the circuit is Boolean and if the input is a tuple of arbitrary inputs (messages) in the field, then we assume an arithmetic circuit.

We consider circuits as algorithms and use them as an alternative approach to the traditional Turing machine model of computation.[2] Notice that any given circuit, C, can only compute on inputs of the same length whereas a Turing machine M takes inputs of any size n. However, a circuit always halts on a given input whereas a Turing machine may not. For the purpose of our discussion relating to cryptography, we assume every Turing machines halts unless otherwise stated. To allow circuits to handle arbitrary length inputs we consider families of circuits.

Definition 2 (Circuit family [**MF21**])**.** A circuit family $C = \{C_n\}_{n\in\mathbb{N}}$ is an indexed set of circuits $C_n \colon \mathbb{F}^{n+r} \to \mathbb{F}^m$ where $r, m = \text{poly}(n)$.

For any input x with length n, $C(x) \stackrel{\text{def}}{=} C_n(x)$. For each circuit $C_n \in C$, r represents the random coins used. If $r = 0$ for all n then C is a deterministic circuit family. A circuit family is said to have polynomial-size if there exists a polynomial p such that $|C_n| < p(n)$ for all n.

Complexity classes

Definition 3 (Complexity Class P)**.** P is the set of languages $\mathscr{L}$ such that there exists a deterministic polynomial-time Turing machine M satisfying $M(x) = 1 \iff x \in \mathscr{L}$

Definition 4 (Complexity Class BPP)**.** BPP is the set of languages $\mathscr{L}$ such that there exists a probabilistic polynomial-time (PPT) Turing machine M satisfying

$$\Pr[M(x) = 1] \geq 2/3 \text{ if } x \in L$$
$$\Pr[M(x) = 1] \leq 1/3 \text{ if } x \notin L$$

Definition 5 (Complexity Class P/poly)**.** P/poly is the set of languages $\mathscr{L}$ such that there exists a polynomial-size circuit family C satisfying $C(x) = 1 \iff x \in \mathscr{L}$

Informally speaking, circuit families is a stronger model of computation than the PPT Turing machine model in the sense that if there exists a PPT Turing machine for deciding

converting every gate to XOR and AND gates and using $\text{XOR}(a, b) = a + b - 2 * a * b$, $\text{AND}(a, b) = a * b$.

[2]The reason for this alternative model is to assume adversaries are computationally "stronger". See Theorem **??**.

a problem, then there also exists a circuit family that can decide the same problem. The formal statement is as follows:

Theorem 1. $\mathrm{P} \subseteq \mathrm{BPP} \subsetneq \mathrm{P/poly}$

The first inclusion follows from the fact that if there exists a deterministic Turing machine that decides a language, then that same machine can be seen as a PPT machine that ignores a given input sequence of coin tosses. For the second inclusion, consider a language $\mathscr{L} \in \mathrm{BPP}$ and a corresponding PPT machine M for $\mathscr{L}$. Then, for any given input x_n with length n, at least $2/3$ of the set of all possible coin toss sequences are good (good r means $M(x_n; r) = 1 \iff x_n \in \mathscr{L}$). This means that there exists at least 1 sequence of coin tosses that yields the correct result for $2/3$ of the possible inputs of length n. By using an alternative (but equivalent) definition of BPP, it can be shown that there actually exist a coin toss sequence, r_n, that yields the correct result for all inputs of length n (see [**Adleman1978TwoTO**; **Gol01**] for more detail). Consider circuit $C_n \colon \{0,1\}^{n+|r_n|} \to \{0,1\}$ with r_n hardcoded as inputs where C_n simulates M using r_n, i.e., $C_n(x_n) = M(x; r)$. Therefore $C_n(x) = 1 \iff x \in \mathscr{L}$ and C decides $\mathscr{L}$.

Interestingly the first inclusion is speculated to be set equivalence [**Arora**], meaning that a deterministic machine could decide the same languages as a probabilistic one. The second inclusion is proper since every unary language is in P/poly whereas undecidable ones are not in BPP (see [**Arora**] for details). In this sense, circuit families are a stronger model of computation than PPT Turing machines. We capture this notion with uniformity.

Definition 6 (Uniform circuit family)**.** A circuit family $\{C_n\}_{n \in \mathbb{N}}$ is uniform if there exists a polynomial-time Turing machine M such that $M(1^n)$ outputs the description of C_n for all $n \in \mathbb{N}$.

A uniform circuit family is polynomial size. The converse is not necessarily true. A family that is not uniform is said to be a non-uniform circuit family. Note that Turing machines are at least as strong as uniform circuit families. More formally, if a uniform circuit family decides $\mathscr{L}$ then there exists a polynomial-time Turing machine that decides $\mathscr{L}$.[3] Simply construct the polynomial-time Turing machine that given any input $x \in \mathscr{L}$, first generates a description of $C_{|x|}$ and then simulates $C_{|x|}(x)$. In other words, the non-uniform circuit families are stronger than the polynomial-time Turing machines.

[3]The converse is also true, meaning deterministic polynomial-time Turing machines are exactly as powerful as uniform circuit families. See [**Arora**] for details

1.2 Probability Theory

Definition 7 (Probability ensemble)**.** Let I be a countable index set. A *probability ensemble* indexed by I (or just ensemble) is a sequence of random variables $(X_i)_{i\in I}$.

In this paper we will exclusively use $\mathbb{N}$ as the index set. For example, the encryption function is a random variable (PPT algorithm), meaning that a single message m corresponds to many valid ciphertexts. By varying the security parameter of the scheme we construct the ensemble $\{Enc(pk, m)\}_{pk\in\mathbb{N}}$

Definition 8 (Statistical distance of ensembles)**.** Let S_n be finite set for all $n \in \mathbb{N}$ and let $X = (X_n \colon \Omega_n \to S_n)_{n\in\mathbb{N}}$, $Y = (Y_n \colon \Omega_n \to S_n)_{n\in\mathbb{N}}$ be two ensembles. The *statistical distance* is defined as

$$\Delta_{X,Y}(n) \stackrel{\text{def}}{=} \frac{1}{2} \sum_{\alpha\in S_n} |\Pr[X_n = \alpha] - \Pr[Y_n = \alpha]|.$$

Definition 9 (Negligible function)**.** Let $f \colon \mathbb{N} \to [0, 1]$. f is negligible if for all positive polynomials $p(\cdot)$, there exists an N such that for all $n > N$, $f(n) < \frac{1}{p(n)}$. We say $f = \text{negl}(n)$.

Definition 10 (Perfectly indistinguishable)**.** Two ensembles X and Y are *perfectly indistinguishable* if $\Delta_{X,Y}(n) = 0$ for all n.

Definition 11 (Statistically indistinguishable)**.** Two ensembles X and Y are *statistically indistinguishable* if $\Delta_{X,Y}(n) = \text{negl}(n)$.

A desirable property of encryption schemes is to make it unfeasible for adversaries to distinguish encryptions. For two random variables X and Y, we want every adversary C to not be able to distinguish samples from X and Y. More precisely, an adversary tasked with identifying whether given samples are from X (C outputs 0) or from Y (C outputs 1) should have roughly the same probability of success irrespective of which random variable the samples are generated from. The formal definition is as follows:

Definition 12 (Computationally indistinguishable [**Gol01**])**.** Two ensembles X and Y are computationally indistinguishable if, for every polynomial-size circuit family $C = \{C_n\}_{n\in\mathbb{N}}$,

$$\text{Adv}_{X,Y}(n) \stackrel{\text{def}}{=} |\Pr[C(X_n) = 1] - \Pr[C(Y_n) = 1]| = \text{negl}(n).$$

The randomness is taken over the outcomes of the random variables X_n and Y_n.

$C(X_n), C(Y_n)$ means that the adversary has access to random oracle returning a sample from X_n and Y_n respectively. The point is that, for every adversary C guessing the samples are from one of the random variables (1 in this case represents from Y_n), the probability of being correct is, up to negligible error, equal to the probability of being incorrect. For example, an adversary always guessing 1 has 100% probability of being incorrect when given samples from X_n and 100% probability of being correct when given samples from Y_n. The intuition is that the advantage in distinguishing the random variables in the ensembles goes to zero fast (faster than the inverse of all polynomials).

Perfect distinguishability implies statistical indistinguishability as 0 is negligible. Statistical indistinguishability implies computational indistinguishability. To see why, assume $\Delta_{X,Y}(n) = \text{negl}(n)$ and consider any circuit family C. Since X_n and Y_n are defined on the same sample space we have

$$\begin{aligned}
\text{Adv}_{X,Y}(n) &= |\Pr(X_n) = 1] - \Pr[C(Y_n) = 1]| \\
&\leq \sum_{\alpha \in S_n} |\Pr[C(X_n = \alpha) = 1 \mid X_n = \alpha] \Pr[X_n = \alpha] \\
&\qquad - \Pr[C(Y_n = \alpha) = 1 \mid Y_n = \alpha] \Pr[Y_n = \alpha]| \\
&\leq \sum_{\alpha \in S_n} |\Pr[X_n = \alpha] - \Pr[Y_n = \alpha]| = 2\Delta_{X,Y}(n) = \text{negl}(n)
\end{aligned}$$

where the first inequality is due to the triangle inequality and the second is due to conditional probability being upper bounded by 1. Intuitively, there does not exist enough distinguishable outcomes to distinguish the ensembles even if there exists an adversary that always correctly recognize an outcome from X and Y respectively.

Homomorphic encryption schemes are based on noise. As we will see in Subsection **??**, noise can be sampled from discrete spaces in accordance with a distribution. Since distributions are central in cryptography, it is important they are understood. A distribution is a probability measure on a measurable space $(S, \mathcal{S})$. Typically, probability distributions are associated with random variables; however, in the absence of random variables, distributions are understood as a specified measure function on the given measurable space (See [**kallenberg-probability**]).

Definition 13 (Discrete distribution measure of a random variable)**.** Consider a

probability space $(\Omega, \mathcal{F}, \Pr)$ and a discrete measurable space $(S, \mathcal{S})$. Let $X\colon \Omega \mapsto S$ be a random variable. The discrete distribution measure of X, or simply *distribution* of X, χ is defined as follows

$$\chi\colon \mathcal{S} \to [0,1]$$
$$\{x\} \mapsto \Pr[X = x]$$

We say that χ is the distribution or*law* of X, denoted $\mathcal{L}(X)$.

Remark. Since χ is a measure on a discrete space, the description of the singletons are sufficient. More explicitly, $\chi(A) = \sum_{x \in A} \Pr[X = x] \quad A \in \mathcal{S}$

Remark. Two different random variables can have the same distribution and one random variable can have two different distributions. See [**cont-finance**] for a formal and excellent discussion.

A distribution measure for a random variable is a probability measure on the sample space $(S, \mathcal{S})$, as opposed to on the outcome space $(\Omega, \mathcal{F})$. For a given probability space, any random variable X defined on it has a distribution associated with it. We write $x \leftarrow X$ or $x \leftarrow \chi$ for sampling the outcome x from X assuming χ is its distribution. When X is a space we mean that x is uniformly sampled from X.

Definition 14 (Statistical distance of distributions)**.** Let χ_1, χ_2 be two discrete distributions (i.e., probability measures) on the measurable space $(S, \mathcal{S})$. The statistical distance is defined as

$$\Delta(\chi_1, \chi_2) \stackrel{\text{def}}{=} \frac{1}{2} \sum_{x \in S} |\chi_1(\{x\}) - \chi_2(\{x\})|$$

Remark. This definition assumes every singleton is measurable. A more general definition is $\Delta(\chi_1, \chi_2) \stackrel{\text{def}}{=} \sup_{A \in \mathcal{S}} |\chi_1(A) - \chi_2(A)|$

Add a part about discrete gaussian distribution

1.3 Cryptographic primitives

In the following definitions we let the message space be denoted $\mathcal{X}$, the ciphertext space be denoted $\mathcal{Y}$, and the key space be denoted $\mathcal{K} = \mathcal{K}_{pk} \times \mathcal{K}_{sk}$.

Definition 15 (Encryption scheme)**.** A correct asymmetric encryption scheme $\mathcal{E} =$ (KeyGen, Enc, Dec) is a triple of algorithms satisfying the following:

- KeyGen : $\{1\}^* \to \mathcal{K}$ is PPT given by $1^\lambda \mapsto (pk, sk)$.
- Enc : $\mathcal{K}_{pk} \times \mathcal{X} \to \mathcal{Y}$ is PPT given by $(pk, m) \mapsto c$.
- Dec : $\mathcal{K}_{sk} \times \mathcal{Y} \to \mathcal{X}$ is deterministic and satisfies $(pk, sk) \leftarrow \text{KeyGen}(1^\lambda) \implies \text{Dec}(sk, \text{Enc}(pk, m)) = m$.

Remark. This paper is only concerned with encryption schemes where decryption always works (correct) and where more than one key is used (asymmetric). Thus, every encryption scheme is assumed to be a correct asymmetric encryption scheme. Furthermore, the decryption algorithm can be considered a PPT algorithm that with probability 1 outputs the correct message for the correct secret key. In other words, the algorithm ignores the input coin toss sequence.

In this paper, we consider homomorphic encryption (HE) schemes. These schemes include a fourth algorithm, Eval, called the evaluation algorithm which is used by the server calculating on encrypted data.

Definition 16 ($\mathcal{C}$-homomorphic encryption scheme)**.** An encryption scheme $\mathcal{E}$ is a $\mathcal{C}$-homomorphic encryption scheme for the set of circuits $\mathcal{C}$ if there exists an extra algorithm Eval such that for any $C \in \mathcal{C}$ taking t inputs the following condition holds:

- Eval: $\mathcal{K}_{pk} \times \mathcal{C} \times \mathcal{Y}^* \to \mathcal{Y}$ is PPT and satisfies $(pk, sk) \leftarrow \text{KeyGen}(1^\lambda) \implies \text{Dec}(sk, \text{Eval}(pk, C, \langle \text{Enc}(pk, m_1), \ldots, \text{Enc}(pk, m_t) \rangle)) = C(m_1, \ldots, m_t)$

We say $\mathcal{E}$ can evaluate all circuits in $\mathcal{C}$ and is $\mathcal{C}$-homomorphic.

The evaluation algorithm runs the ciphertexts through the permissible circuit while also satisfying the requirement that decrypting the resulting ciphertext yields the same result as the plaintexts running through the circuit. To its disposal, the evaluation algorithm is given the public key. The ciphertexts returned by the Eval algorithm are called *evaluated ciphertexts* (suggesting the circuit has evaluated the ciphertexts) and those returned by the encryption algorithm are called *fresh ciphertexts*. Remark that correctness is only guaranteed if the Eval algorithm is given fresh ciphertexts. If the circuit corresponds to the computable function f acting on a vector c of ciphertexts, we denote the evaluated ciphertexts c_f (i.e., $c_f \stackrel{\text{def}}{=} \text{Eval}(pk, f, c)$). Similarly, for the vector m of plaintexts, we denote the evaluated plaintexts m_f (i.e., $m_f \stackrel{\text{def}}{=} f(m)$). Thus, the condition for the Eval algorithm can be written $(pk, sk) \leftarrow \text{KeyGen}(1^\lambda) \implies \text{Dec}(sk, c_f) = m_f$. In a homomorphic encryption scheme that supports one addition and multiplication of

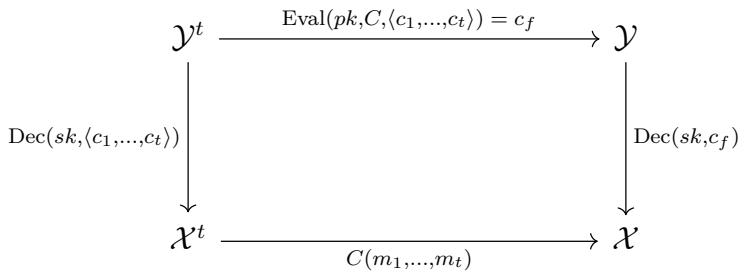

Figure 1: The decryption homomorphism. The path through $\mathcal{Y}$ represents computing before decrypting. The path through $\mathcal{X}^t$ represents decrypting before computing. Credit: [**Gen14-edge**].

fresh ciphertexts, the decryption function is a ring homomorphism. Consider a valid key pair (sk, pk) which are, for notational simplicity, hard-wired into the decryption and evaluate functions respectively, plaintext ciphertext pairs $(m_1, c_1 = \mathrm{Enc}_{pk}(m_1))$, $(m_2, c_2 = \mathrm{Enc}_{pk}(m_2))$ and circuits $C_+(m_1, m_2) \stackrel{\text{def}}{=} m_1 + m_2$ and $C_\times(m_1, m_2) \stackrel{\text{def}}{=} m_1 \times m_2$ that can be evaluated by the scheme.

$$\begin{aligned}
\mathrm{Dec}_{sk}(c_1 + c_2) &= \mathrm{Dec}_{sk}(\mathrm{Eval}_{pk}(C_+, \langle c_1, c_2 \rangle) = C_+(m_1, m_2) = m_1 + m_2 \\
&= \mathrm{Dec}_{sk}(c_1) + \mathrm{Dec}_{sk}(c_2) \\
\mathrm{Dec}_{sk}(c_1 \times c_2) &= \mathrm{Dec}_{sk}(\mathrm{Eval}_{pk}(C_\times, \langle c_1, c_2 \rangle) = C_\times(m_1, m_2) = m_1 \times m_2 \\
&= \mathrm{Dec}_{sk}(c_1) \times \mathrm{Dec}_{sk}(c_2)
\end{aligned}$$

The main idea behind a homomorphic encryption scheme is to give a server encrypted data so that it can compute on that data and return the answer in encrypted form. However, the definition provided allows for trivial homomorphic encryption schemes where the server does nothing. More specifically, consider any set of circuits $\mathcal{C}$ and let $\mathrm{Eval}(pk, C, \langle c_1, \ldots, c_t \rangle) = (C, \langle c_1, \ldots, c_t \rangle)$. Eval takes a description of a circuit and a tuple of ciphertexts, one for each input wire of the circuit, and simply outputs the description of the circuit together with the given tuple. Clearly, Eval runs in polynomial time. Consider a decryption algorithm that, if given an input of this form, first decrypts the t ciphertexts and then computes m_f using C. To ensure that the server actually processes the given inputs we introduce compactness.

Definition 17 (Compactness)**.** A $\mathcal{C}$-homomorphic encryption scheme is compact if there exists a polynomial $p(\lambda)$ such that for all $(pk, sk) \leftarrow \mathrm{KeyGen}(1^\lambda)$, for every $C \in \mathcal{C}$ taking any number t inputs and any $c \in \mathcal{Y}^t$, the size of the output $\mathrm{Eval}(pk, C, \langle c_1, \ldots, c_t \rangle)$ is less than $p(\lambda)$. We say that the scheme compactly evaluates $\mathcal{C}$.

For a compact $\mathcal{C}$-homomorphic encryption scheme, the size of the output is independent of the circuit function used. In particular, the previous, trivial homomorphic encryption scheme where $\text{Eval}(pk, C, \langle c_1, \ldots, c_t \rangle) = (C, \langle c_1, \ldots, c_t \rangle)$ is not compact for any set of circuits with unbounded circuit size, which includes circuit families with circuits that do not ignore all except for constantly many inputs, meaning essentially every application of a HE scheme.

Definition 18 (Fully Homomorphic Encryption (pure FHE))**.** Let $\mathcal{C}$ be the class of all circuits. An encryption scheme $\mathcal{E}$ is a fully homomorphic encryption (pure FHE) scheme if it is $\mathcal{C}$-homomorphic and compactly evaluates $\mathcal{C}$.

For a scheme to be fully homomorphic it is required that it can evaluate circuits of arbitrary size. Many times it suffices to consider only circuits of a beforehand specified depth, L, as any deeper circuits are irrelevant to the application. The following definition capture schemes that can evaluate any set of circuits with depths bounded by the client.

Definition 19 (Leveled fully homomorphic encryption (leveled FHE))**.** An encryption scheme $\mathcal{E}$ with the KeyGen algorithm modified is a leveled fully homomorphic encryption scheme if it satisfies the following:

- $\text{KeyGen} : \{1\}^* \times \{1\}^* \to \mathcal{K}$ is PPT given by $(1^\lambda, 1^L) \mapsto (pk, sk)$.
- Let $\mathcal{C}_L$ be the set of circuits with depth less than or equal to L. Then $\mathcal{E}$ is $\mathcal{C}_L$-homomorphic.
- $\mathcal{E}$ compactly evaluates the set of all circuits.

Remark. Notice that the length of the evaluated ciphertexts in a leveled FHE scheme is independent of the depth. Also note that for any circuit C, there exists an L such that a leveled FHE scheme can evaluate it.

A potential point of contention is the purpose of a pure FHE scheme in the presence of a leveled FHE scheme; why do we not simply choose an L such that the leveled FHE scheme can evaluate any given circuit C?

Possible answer? ask Johan

It's not always clear what the depth of the circuit is. Therefore, it may be needed to use an excessively large depth parameter L. In a leveled scheme, an increasing L allows for longer keys and ciphertext lengths and consequently a performance overhead. In

contrast, a pure FHE scheme is independent of the depth, meaning ciphertexts lengths are only bounded by the security parameter.

Security definitions

In this paper, semantic security refers to security against chosen-plaintext attack (CPA). The definition relates to the following game where the challenger possess the secret key and the player is the adversary trying to break the scheme. Consider encryption scheme (KeyGen, Enc, Dec, Eval) and polynomial-size Boolean circuit family $C = \{C_n\}_{n\in\mathbb{N}}$. The CPA game is defined with the Boolean function $\text{CPA}_C(\lambda)$ as follows:

1. **Setup**: Challenger samples $pk \leftarrow \text{KeyGen}(1^\lambda)$ and sends it to player.
2. **Choose**: Player C selects two distinct plaintext messages $(m_0, m_1) \leftarrow C(pk)$ of the same length, and sends them to the challenger.
3. **Encrypt**: The challenger randomly picks a bit $b \in \{0, 1\}$ and encrypts the message m_b. The encrypted message $c \stackrel{\text{def}}{=} \text{Enc}(pk, m_b)$, called challenge ciphertext, is sent to the player.
4. **Guess**: Player C output guess $b' \in \{0, 1\}$.
5. **Win**: $\text{CPA}_C(\lambda) = \begin{cases} 1 & \text{if } b = b' \\ 0 & \text{if } b \neq b'. \end{cases}$

If $\text{CPA}_C(\lambda) = 1$ then the adversary C guessed correctly which of their two chosen messages was encrypted, based only on observing the ciphertext. Notice that the game requires the player to choose messages of equal length in the 'choose' phase since the ciphertext length always leaks information about the length of the message, namely an upper bound on the message length.

Definition 20 (Semantic security (CPA))**.** An encryption scheme is semantically secure if, for all polynomial-size Boolean circuit families C,

$$|\Pr[\text{CPA}_C(\lambda)] - \frac{1}{2}| = \text{negl}(\lambda).$$

Semantic security means that there exists no algorithm in P/poly that can do more than negligibly better than guessing randomly in determining the message. Semantic security is equivalent to indistinguishability of encryptions (see [**Gol04**] for proof).

Definition 21 (Indistinguishability of encryptions)**.** An encryption scheme has indistinguishable encryptions if for any key $(pk, sk) \leftarrow \text{KeyGen}(1^\lambda)$ and any two distinct messages m_1, m_2 of equal length, the ensembles $\{\text{Enc}(pk, m_1)\}_{\lambda\in\mathbb{N}}$ and $\{\text{Enc}(pk, m_2)\}_{\lambda\in\mathbb{N}}$ are computationally indistinguishable.

Usually, encryption schemes are required to be secure against a stronger type of attack, called chosen-ciphertext attack (CCA). There are two types of CCA attacks; adaptive (called CCA2) and non-adaptive (called CCA1). The CCA1 game is defined exactly like the CPA game but where the player also has oracle access to the decryption algorithm in the choose phase. In other words, the player can decrypt any ciphertext of their choice before submitting the two messages m_0 and m_1 to the challenger. The CCA2 game is the same as CCA1 except that the player also has oracle access to the decryption algorithm in the guess phase for every ciphertext except the challenge ciphertext. Security against CCA1 and CCA2 attacks are defined analogously to semantic security. Clearly, CCA2 security implies CCA1 security and CCA1 security implies semantic security.

As a consequence of its design, homomorphic encryption schemes cannot be CC2 secure. The reason is that the player can run the evaluate algorithm on the challenge ciphertext with any circuit of choice and then decrypt the evaluated ciphertext. More formally, consider any challenge ciphertext $\text{Enc}(pk, m_b)$ and the identity circuit C. Is the identity circuit always in C?. Player runs $\text{Eval}(pk, C, \text{Enc}(pk, m_b))$, generating a valid evaluated ciphertext of $C(m_b)$. Player then queries decryption and yields $C(m_b) = m_b$. Homomorphic encryption schemes allow the attacker to transform the ciphertext of a message m to a ciphertext to a message related to m by a known function. This property is called *malleability*.

Definition 22 (Circuit privacy)**.**

One last security definition that will be relevant later is circular security

Definition 23 (Circular security)**.** A semantically secure homomorphic encryption scheme is circular secure if it is semantically secure even when the adversary is given encryptions of the secret key.

Remark. Circular security is not implied by semantic security because an adversary with a random access oracle cannot efficiently query encryptions of the secret key [**Bra18-survey**].

1.4 Lattices

In this section, we assume every vector is a column vector and we denote a matrix $\mathbf{A}$ with boldface. By $\mathbf{A} \in \mathbb{F}^{n\times m}$ we mean $\mathbf{A}$ is a $n \times m$ matrix with entries in $\mathbb{F}$. Consider a basis $\mathbf{B} = \{b_1, \ldots, b_k\}$. A lattice with basis $\mathbf{B}$ is defined $\mathcal{L}(\mathbf{B}) \stackrel{\text{def}}{=} \{\sum_{i=1}^{k} a_i b_i \mid a_i \in \mathbb{Z}\}$. For any integer $q \geq 2$, we define $\mathbb{Z}_q \stackrel{\text{def}}{=} (-\frac{q}{2}, \frac{q}{2}] \cap \mathbb{Z}$. For any tuple of integers x (e.g., integer or matrix), we define $[x]_q$ as the tuple of integers over $\mathbb{Z}_q$ such that each element is congruent mod q to the corresponding element in x. For example, $q = 3, x = (5, 1, -2, 6), [x]_q = (-1, 1, 1, 0)$.

LWE and RLWE

This section is based on [**Hal18**], [**LNP22**] and [**Pei16-decade**].

The LWE problem is to solve a system of linear equations with a small noise added. The parameters for the LWE problem are the integers n for the dimension, $q = q(n)$ for the modulus, m for number of samples and a discrete error distribution measure χ on $\mathbb{Z}_q$.

The requirement is $q(n) = \Omega(n)$(Is this true?), $m = \Theta(n \log q)$ and $\Pr[|x| \geq aq \mid x \leftarrow \chi] = negl(n)$. I think this is requirement for hardness, not for the problem itself. This means that as the dimension grows and q also grows, each component of the error grows (assuming a doesnt decrease fast than q)

Consider the space of $n \times m$ matrices $\mathbb{Z}_q^{n\times m}$ with uniform distribution μ, $\mathbb{Z}_q^m$ with discrete distribution χ^m, the probability space $(\mathbb{Z}_q^{n\times m} \times \mathbb{Z}_q^m, \mathcal{P}, \mu \otimes \chi^m)$ equipped Explain why using power set with power set $\mathcal{P}$ and product measure $\mu \otimes \chi^m$, a uniformly random[4] $s \in \mathbb{Z}_q^n$ and the random variable $A_{s,n,q,\chi,m}$ defined on the probability space as follows

$$\begin{aligned} A_{s,n,q,\chi,m} \colon \mathbb{Z}_q^{n\times m} \times \mathbb{Z}_q^m &\to \mathbb{Z}_q^{n\times m} \times \mathbb{Z}_q^m \\ (\mathbf{A}, e) &\mapsto (\mathbf{A}, s^T\mathbf{A} + e^T) \end{aligned}$$

Definition 24 (LWE distribution)**.** The learning with errors distribution is defined as the distribution of $A_{s,n,q,\chi,m}$

$$\text{LWE}_{s,n,q,\chi,m} \stackrel{\text{def}}{=} \mathcal{L}(A_{s,\chi})$$

[4]Secret s can also be chosen with respect to distribution χ^n. See [**Applebaum**].

Remark. There are many LWE distributions, each characterised by the the parameters n, m, q, s, χ.

In words, first fix a secret s with entries in $\mathbb{Z}_q$ at random. Then sample a $n \times m$ matrix $\mathbf{A}$ with entries in $\mathbb{Z}_q$ at random and e consisting of m independent errors from the discrete distribution χ. Calculate $s^T\mathbf{A} + e^T$ and output the pair $(\mathbf{A}, s^T\mathbf{A} + e^T)$. The LWE distribution specifies the probability of sampling each pair.

Note that the LWE distribution is not the same as the product measure in the probability space $A_{s,\chi}$ is defined on.

Definition 25 (search-LWE problem)**.** Let $s \leftarrow \mathbb{Z}_q^n$ be random. Let $(\mathbf{A}, b) \leftarrow \text{LWE}_{s,n,q,\chi,m}$. The (average-case) search-LWE problem is to construct a PPT algorithm A such that

$$\Pr[A(\mathbf{A}, b) = s] \neq \text{negl}(n) \tag{1.1}$$

Definition 26 (decision-LWE problem)**.** Let $s \leftarrow \mathbb{Z}_q^n$ be random. Let $\mathcal{U}_n$ be the uniform distribution on $\mathbb{Z}_q^{n \times m} \times \mathbb{Z}_q^m$. The (average-case) decision-LWE problem is to distinguish $\mathcal{U}_n$ and $\text{LWE}_{s,n,q,\chi,m}$ with non negligible advantage w.r.t n Connect to ensembles being computationally indistinguishable

Remark. Since the requirement is that s is randomly chosen instead of arbitrarily chosen, the problem refers to average-case hardness as opposed to worst-case hardness. This is actually equivalent, meaning if average keys are easy to guess, then all keys are easy to guess. See [**Reg10**] for proof.

The LWE search and decision problem are equivalent when q is prime and $q(n) = \text{poly}(n)$.

1.5 History of fully homomorphic encryption

Began with rivest, adleman and Dertouzos in 1978 and the paper about Data Banks. They identified the use of delegating computation to a third party.

To perform homomorphic evaluation on arbitrary functions, a scheme needs to support unlimited number of multiplication and addition of ciphertexts. The problem was thus to find a scheme that both secure and also supports homomorphic addition and multiplication.

RSA supports multitplication,

Paillier supports addition.

1.5.1 First generation

Gentry scheme is based on ideal lattices. The construction considers a polynomial ring over the integers modulo an ideal generated by a cyclotomic polynomial. Then there exists an ideal with norm polynomially bounded with respect to the dimension such that the ideal generates a lattice. However, his decryption function was not bootstrappable. He used a method called squashing to make it so. The idea was to include some extra information in the public key which allowed for easier decryption but introduced hardness assumption w.r.t sparse subset sum problem.

The first generation was improved by Smart and Vercauteren in [**SV09-batch**] where they introduced batching of multiple plaintexts into a single ciphertext using the Chinese Remainder Theorem. Gentry and Halevi implemented Gentry's scheme in [**GS10-impl**], including the use of the batching technique and an optimized squashing technique to bring down the degree of the decryption polynomial from hundreds (estimated by Smart and Vercauteren) to 15. In Gentry's original scheme, the key generation algorithm was impractically slow. In their implementation, they reduced the asymptotic complexity to $\tilde{O}\left(n^{1.5}\right)$ for cyclotomic fields where the root of unity is a power of 2. Scholl and Smart extended the implementation to arbitrary cyclotomic fields in [**SS11-keygen**].

At the end of 2009, Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan introduced a new scheme that was based on

First generation: Started with Gentry's thesis. The first generation includes: 1. Gentrys scheme based on ideal lattices. 2. Simplification to integers 3. First implementation by gentry and halevi. 4. Uses mod-p number and arithmetic circuits.

Second generation: Started with BV11 paper. The second generation includes: 1. Basing hardness on LWE instead of ideal lattice problems (BV11) 2. Modulus switching for noise management (BV11) 3. SIMD operation 4. Uses mod-p number and arithmetic circuits.

Third generation: Started with GSW13 paper. The third generation includes: 1. Improvements. Some parameter choices made noise growth slow. 2. Tweak of GSW allowed for efficient bootstrapping technique, by BV. More bootstrapping optimizations

introduced and now it's 0.1 seconds. 3. Third generation does not allow for SIMD operations (ciphertext packing). 4. Uses bits and boolean circuits.

Fourth generation: Started with CKKS16 paper. The fourth generation includes: 1. Used on floating point numbers (real and complex). Useful for ML, NN. Fourth generation is based on CKKS scheme. This scheme allows for computation on real and complex numbers. This is useful for machine learning applications.

Write about Regev and GSW. See gentry 14 survey.

1.6 Noise management

The main problem with homomorphic encryption schemes is the noise. Noise is introduced in the ciphertexts during the encryption process and when the ciphertexts undergo homomorphic operations, the noise grows. After sufficiently many operations, the noise grows to the point where the decryption of the evaluated ciphertext fails. In order to reach FHE, it is necessary to control the noise to allow for sufficient number of operations. Noise management is the process of controlling the noise during homomorphic evaluation. In his 2009 seminal PHD thesis [**Gentry-Thesis**], Craig Gentry showed that FHE was possible by using a noise management technique called Bootstrapping. Bootstrapping is an algorithm that transforms a possibly noisy ciphertext into a correct evaluated ciphertext with little noise by using an encryption of the secret key to decrypt the noisy ciphertext homomorphically.

Evaluated ciphertexts vs fresh ciphertexts

Recall that in Definition **??**, the decryption of an evaluated ciphertext is only required to be correct if the ciphertext is a fresh ciphertext. Explain Why we care. The question is then if it is possible to pass evaluated ciphertext to the Eval algorithm instead of fresh ones.

A HE scheme is said to be i-hop if the Eval algorithm can be called on its own output up to i times. A HE scheme that satisfies the definition is 0-hop, also called *single-hop*, and multi-hop if it is i-hop for all i.

Note on hard-wiring ciphertexts in circuits

Not nessesary. Gentry did not hardwire, and instead he encrypted messages twice, passed the double encrypted message as inout where the inner layer was decrypted homomorphically. In practice, the server sees the ciphertext and constructs the dec function for that ciphertext specifically, designed to only take the secret key as input.

Key-Switching

This section is based on [**Bra18-survey**].

As a natural segway into bootstrapping, we first introduce a related but different concept called key-switching. Since HE schemes are designed with the constraint of supporting homomorphic operations, they are often inefficient. Therefore, it would be desirable to use a more efficient non-homomorphic scheme to encrypt the large messages, but still retaining the homomorphic property. Key-switching is a technique that allows for homomorphic computation on ciphertexts encrypted under a non-homomorphic scheme. In particular, it allows for transforming a ciphertext under a non-HE scheme to a corresponding ciphertext under a HE scheme. The idea is to encrypt the much shorter secret key of the non-HE scheme under the public key of the HE scheme and hard-wire the ciphertexts into the function of interest.

Let (H.KeyGen, H.Enc, H.Dec, Eval) be a homomorphic encryption scheme and (KeyGen, Enc, Dec) be a non-homomorphic encryption scheme. Let $(hpk, hsk) \leftarrow \mathrm{H.KeyGen}(\lambda)$ and $(pk, sk) \leftarrow \mathrm{KeyGen}(\lambda)$. Consider computable function C, the vector of ciphertexts $c \leftarrow \mathrm{Enc}(pk, m)$ and an encrypted secret key $sk' \leftarrow \mathrm{H.Enc}(hpk, sk)$. Define $\hat{C}_c(\cdot) \stackrel{\text{def}}{=} C(\mathrm{Dec}(\cdot, c))$. $C(m)$ can be computed homomorphically by decrypting $\mathrm{Eval}(hpk, \hat{C}_c, sk')$. Indeed, this is a correct encryption of $C(m)$ since

$$\mathrm{Dec}(hsk, \mathrm{Eval}(hpk, \hat{C}_c, sk')) = \hat{C}_c(sk) = C(\mathrm{Dec}(sk, c)) = C(m)$$

We have shown that it is possible to use a non-homomorphic encryption scheme to encrypt the message and still perform homomorphic computation on it. Key-switching transforms a ciphertext encrypted under a non-HE scheme to an evaluated ciphertext encrypted under a HE scheme.

The underlying assumption is that the HE scheme can evaluate $\hat{C}_c$, meaning it can first

run the decryption algorithm and then compute the desired function C to generate a valid evaluated ciphertext. Even under the assumption that the decryption algorithm is simple enough to be evaluated by the scheme, a general C consists of several multiplication and addition gates, meaning it is unlikely that the scheme can evaluate $\hat{C}_c$. However, if it is possible to split C into smaller components, $C = C^m \circ \cdots \circ C^1$, and evaluate each component separately, $c_i = \text{Eval}(hpk, \hat{C}^i_{c_{i-1}}, sk')$, where $c_0 \leftarrow Enc(pk, m)$, computing C homomorphically is achievable assuming $\hat{C}^i_{c_{i-1}}$ is permissible for all i. However, as the construction currently stands, further computation on the ciphertext is not possible. To see why, consider $c_1 = \text{Eval}(hpk, \hat{C}^1_{c_0}, sk')$ and let $c_2 = \text{Eval}(hpk, \hat{C}^2_{c_1}, sk')$. We hope decrypting c_2 yields $(C^2 \circ C^1)(m)$, but $\text{Dec}(hsk, c_2) = \text{Dec}(hsk, \text{Eval}(hpk, \hat{C}^2_{c_1}, sk')) = C^2(\text{Dec}(sk, c_1))$. However, since c_1 is encrypted under the HE scheme, the non-homomorphic decryption algorithm applied to c_1 does not make sense and decryption fails.

Bootstrapping

Key-switching is a nice optimization technique for encrypting messages faster, but it does not allow for further computation on the ciphertext. The requirement is that the decryption algorithm and the secret key originate from the HE scheme. We therefore only consider the HE scheme. Let (KeyGen, Enc, Dec, Eval) be a HE scheme, $(pk, sk) \leftarrow \text{KeyGen}(\lambda)$ and $sk' \leftarrow \text{Enc}(pk, sk)$. Since this construction encrypts the secret key using its own public key, circular security is assumed (this assumption is discussed further below). Under this construction, decryption of c_1 is still correct as $\text{Dec}(sk, c_1) = C^1_{c_0}(sk) = C^1(\text{Dec}(sk, c_0)) = C^1(m)$, but the difference is that decryption of c_i also works since, by induction,

$$\text{Dec}(hsk, \text{Eval}(hpk, \hat{C}^i_{c_{i-1}}, sk')) = \hat{C}^i_{c_{i-1}}(sk) = C^i(\text{Dec}(sk, c_{i-1})) = C^i \circ C^{i-1} \circ \cdots \circ C^1(m)$$

In essence, we have constructed a method to evaluate C as follows: The first step is to evaluate a component of C on the encryption of the secret key to obtain an evaluated ciphertext. The second step is to generate a circuit which is the composition of the decryption circuit for the previous evaluated ciphertext and the consecutive component of C. The third step is to evaluate the generated circuit on the same ciphertext, namely the encryption of the secret key. Repeating this process ultimately yields c_m which decrypts to $C(m)$. Since each evaluated ciphertext $c_1, \ldots, c_m$ is an evaluation of a fresh

ciphertext, sk', the decryption is guaranteed to work.

Splitting the desired function C into multiple components $C^1, \ldots, C^m$ is the key insight into constructing FHE. Consider the simplest case; each component function is either one multiplication gate or one addition gate. In other words, for a vector of ciphertexts $c = \langle c_1, c_2 \rangle$, define a multiplicative component circuit of C as $C^i(c) = c_1 \times c_2$, $\hat{C}^i_c(\cdot) = C^i(\mathrm{Dec}(\cdot, c)) = C^i(\langle \mathrm{Dec}(\cdot, c_1), \mathrm{Dec}(\cdot, c_2) \rangle) = \mathrm{Dec}(\cdot, c_1) \times \mathrm{Dec}(\cdot, c_2)$ and where the addition case is analogous. Note that $\hat{C}^i_c(sk) = m_1 \times m_2$. It turns out that if the scheme can evaluate just these two types of circuits, then it can evaluate every circuit. This is the idea behind bootstrapping.

Definition 27 (Bootstrappable encryption scheme)**.** Let $\mathcal{E}$ be a $\mathcal{C}$-homomorphic encryption scheme. Consider the following *augmented decryption circuits* for $\mathcal{E}$:

$$B^{(m)}_{c_1,c_2}(\cdot) \stackrel{\text{def}}{=} \mathrm{Dec}(\cdot, c_1) \times \mathrm{Dec}(\cdot, c_2)$$
$$B^{(a)}_{c_1,c_2}(\cdot) \stackrel{\text{def}}{=} \mathrm{Dec}(\cdot, c_1) + \mathrm{Dec}(\cdot, c_2)$$

$\mathcal{E}$ is *bootstrappable* if

$$\{B^{(m)}_{c_1,c_2}(\cdot), B^{(a)}_{c_1,c_2}(\cdot) \mid c_1, c_2 \in \mathcal{Y}\} \subset \mathcal{C}$$

The augmented decryption circuits have the ciphertexts hard-wired into its description and takes as input an encryption of the secret key. A bootstrappable scheme correctly evaluates the set of all augmented decryption circuits. In particular, it correctly evaluates its own decryption circuit by letting c_2 be an encryption of the multiplicative or additive identity respectively.

Theorem 2 (Gentrys bootstrapping theorem, simplified by Vaikuntanathan [**Gentry-Thesis**; **Vai-survey**])**.** *Any bootstrappable scheme is leveled-FHE. Furthermore, if circular security holds, it is pure FHE.*

Remark. Bootstrapping is sufficient for leveled-FHE, but not necessary. See [**BGV12-no-bootstrap**] for a leveled-FHE scheme that does not require bootstrapping.

Proof. test □

We first introduce notation and then explain bootstrapping further. Let m' denote encrypted message m and sk' denote encrypted secret key sk. If a scheme assumes circular

security, then one valid key pair (pk, sk) is sufficient for arbitrarily many subroutine calls where every encryption is made using pk. However, if circular security is not assumed, every bootstrapping subroutine call requires a new valid key pair. Consider a sequence of independently sampled valid key pairs $\{(pk_i, sk_i)\}_{i=0,1,\ldots}$. Let $m^{(k)}$ and $sk_i^{(k)}$ denote valid encryptions under public key pk_k, meaning $\text{Dec}(sk_k, m^{(k)}) = m$ and $\text{Dec}(sk_k, sk_i^{(k)}) = sk_i$. The first bootstrapping subroutine call is done using $i = 0$. In the bootstrapping algorithm we set $k = i+1$ so that the encryption of every secret key made under the next public key, meaning the circular security assumption is avoided.

Showing how a bootstrapping subroutine works

Consider a circular secure bootstrappable homomorphic encryption scheme $\mathcal{E}$ and assume we need the homomorphic multiplication[5] of two ciphertexts m'_1, m'_2 encrypted under pk, but that the noise of the resulting evaluated ciphertext would cause decryption to fail. Consider instead the less noisy ciphertext $m'_{1,2} \stackrel{\text{def}}{=} B^{(m)}_{m'_1,m'_2}(sk')$. In other words, encrypt the secret key and pass it as input to the circuit.

Indeed, $m'_{1,2}$ is a valid ciphertext of $m_1 \times m_2$ since

$$\begin{aligned}
\text{Dec}(sk, m'_{1,2}) &= \text{Dec}(sk, B^{(m)}_{m'_1,m'_2}(sk')) \\
&= B^{(m)}_{m'_1,m'_2}(sk) \\
&= \text{Dec}(sk, m'_1) \times \text{Dec}(sk, m'_2) \\
&= m_1 \times m_2
\end{aligned}$$

If circular security is not assumed, the process becomes more complicated. For bootstrapping round i, where $i = 0$ represents the first round, let $m_{1,2}^{(i+1)} \stackrel{\text{def}}{=} B^{(m)}_{m_1^{(i)},m_2^{(i)}}(sk_i^{(i+1)})$. Correctness follows from

$$\begin{aligned}
\text{Dec}(sk_{i+1}, m_{1,2}^{(i+1)}) &= \text{Dec}(sk_{i+1}, B^{(m)}_{m_1^{(i)},m_2^{(i)}}(sk_i^{(i+1)})) \\
&= B^{(m)}_{m_1^{(i)},m_2^{(i)}}(sk_i) \\
&= \text{Dec}(sk_i, m_1^{(i)}) \times \text{Dec}(sk_i, m_2^{(i)}) \\
&= m_1 \times m_2
\end{aligned}$$

Note that both $m'_{1,2}$ and $m'_1 \times m'_2$ decrypts to $m_1 \times m_2$. The idea behind bootstrapping

[5]The case for addition of ciphertexts works the exact same way. We focus on multiplication here since addition usually only increases the noise slightly.

is that $m'_{1,2}$ has less noise.

In practice, we don't use augmented decryption circuits but instead we just refresh the ciphertext. In this case, we only need the dec func to be permitted.

Contents

Appendix A

First Appendix

This is only slightly related to the rest of the report

Appendix B

Second Appendix

this is the information