

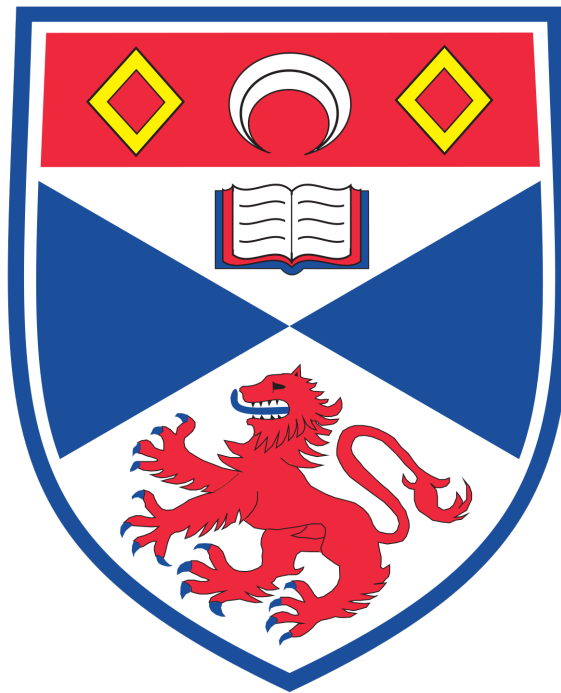
Fully homomorphic encryption

CS4796: Joint Senior Honours Project

Gergely Flamich

Supervisors:

Dr Sophie Huczynska, Prof Steve Linton



School of Computer Science
University of St Andrews
Scotland

Abstract

Declaration We declare that the material submitted for assessment is my own work except where credit is explicitly given to others by citation or acknowledgement. This work was performed during the current academic year except where otherwise stated. The main text of this project report is NN,NNN* words long, including project specification and plan. In submitting this project report to the University of St Andrews, I give permission for it to be made available for use in accordance with the regulations of the University Library. I also give permission for the report to be made available on the Web, for this work to be used in research within the University of St Andrews, and for any software to be released on an open source basis. I retain the copyright in this work, and ownership of any resulting intellectual property.

Contents

1	Introduction	3
2	A Very Brief History of Ciphers	3
3	Perfect Security	4
4	Computational Security	4
5	Public-Key Crypto Schemes	4
5.1	Euler's Theorem	4
5.2	An Example: RSA	4
6	Homomorphic Crypto Schemes	4
6.1	Definition	4
7	Paillier Encryption	4
8	Fully Homomorphic Encryption	4
8.1	Lattices	4
8.2	GGH	4
8.3	Dr Craig Gentry's Scheme	4

1 Introduction

2 A Very Brief History of Ciphers

The history of ciphers goes back at least 2000 years. We will now examine the cipher now named after Julius Caesar, who used the following method to obfuscate his correspondence for his adversaries:

Caesar Cipher : Take the message (written using symbols from the Latin alphabet) that we wish to obfuscate. Replace each symbol with the one that comes 3 places after it in the alphabet. For letters at the end where we could not shift, we “wrap around” to the beginning of the alphabet and carry on counting from there. For example

$$Alea iacta est \rightarrow Dohd mdfzd hhz.$$

To decrypt a message, we simply shift backwards by 3 positions, e.g.

$$Zhqm, zmgm, zmgm \rightarrow Veni, vidi, vici.$$

We can further generalise this concept to arrive at the definition of *shift ciphers*, where instead of the fixing the shift to 3, we pick a key $k \in \mathbb{N}$, and we then shift k places forward (and backwards). With shift ciphers in mind, we now formally defined some key concepts that will be used throughout this work.

Definition 2.1. Cipher: Let $\mathcal{M}, \mathcal{K}_{\text{Enc}}, \mathcal{K}_{\text{Dec}}$ be finite sets of symbols. We will refer to \mathcal{M} as the **message space**. A cipher C over $\mathcal{M}, \mathcal{K}_{\text{Enc}}$ and \mathcal{K}_{Dec} is a tuple $(\text{Gen}, \text{Enc}, \text{Dec})$, where

- **Gen** is the **key generation algorithm**, which outputs a key $(k_{\text{Enc}}, k_{\text{Dec}}) \in (\mathcal{K}_{\text{Enc}}^* \times \mathcal{K}_{\text{Dec}}^*)$, chosen according to some distribution. We refer to the set $\mathcal{K} \subseteq \mathcal{K}_{\text{Enc}}^* \times \mathcal{K}_{\text{Dec}}^*$ of all possible outputs of **Gen** as the **key space** of C . If for all outputs we have $k_{\text{Enc}} = k_{\text{Dec}}$, then we call C a **symmetric cipher** and instead of the tuple we just write k . Otherwise we call C an **asymmetric cipher**.
- **Enc** : $\mathcal{K}_{\text{Enc}}^* \times \mathcal{M} \rightarrow \mathcal{M}$ is the **encryption algorithm**, which takes a encryption key k and a message m and outputs its encoding c . We will often refer to the message as the **plaintext** and to the encrypted message as **ciphertext**.
- **Dec** : $\mathcal{K}_{\text{Dec}}^* \times \mathcal{M} \rightarrow \mathcal{M}$ is the **decryption algorithm**, which takes some decryption key k and some ciphertext c and outputs its corresponding plaintext m .

Finally, we require the correctness of C , concretely

$$\forall m \in \mathcal{M}, \forall (k, k') \in \mathcal{K}. \quad \text{Dec}(k', \text{Enc}(k, m)) = m,$$

i.e. that the cipher does not change its contents.

Let us now revisit shift ciphers over an alphabet $\mathcal{M} = \{m_1, \dots, m_n\}$, and see how they fit our definition of a cipher. First, it is easy to see that we are dealing with a symmetric cipher, and since a shift by k and any $k + xn, x \in \mathbb{N}$ is going to be the same, $\mathcal{K} = \mathbb{Z}_n$. Then,

- **Gen**: **Gen** is uniformly picking an element from \mathbb{Z}_n .

- **Enc:** We note that if we relabel our symbols to their indices (i.e. $m_i \rightarrow i$), then we can express our encryption function as

$$\text{Enc}(k, i) = i + k \pmod n.$$

- **Dec:** Similarly, performing the above relabeling, we get

$$\text{Dec}(k, i) = i - k \pmod n.$$

Now, checking the correctness of C_{shift} : Let $k, m \in \mathbb{Z}_n$

$$\text{Dec}(k, \text{Enc}(k, m)) = m + k \pmod n - k \pmod n = m + k - k \pmod n = m \pmod n = m.$$

Since k and m were arbitrary, it holds $\forall m, k \in \mathbb{Z}_n$.

Kerckhoffs' principle

3 Perfect Security

4 Computational Security

5 Public-Key Crypto Schemes

5.1 Euler's Theorem

5.2 An Example: RSA

6 Homomorphic Crypto Schemes

6.1 Definition

7 Paillier Encryption

Paillier encryption is an additively homomorphic crypto scheme, that operates over \mathbb{Z}_{N^2} where $N = pq$ for some p, q primes. In particular, it takes advantage of the fact that $\mathbb{Z}_{N^2} \simeq \mathbb{Z}_N \times \mathbb{Z}_N^*$.

8 Fully Homomorphic Encryption

8.1 Lattices

8.2 GGH

8.3 Dr Craig Gentry's Scheme