ՀՀ ԳԱԱ ԻՆՖՈՐՄԱՏԻԿԱՅԻ ԵՎ ԱՎՏՈՄԱՏԱՑՄԱՆ ՊՐՈԲԼԵՄՆԵՐԻ ԻՆՍՏԻՏՈՒՏ

Դանոյան Դավիթ Հակոբի

ԱՆՎՏԱՆԳ ԲԱԶՄԱՄԱՍՆԱԿԻՑ ՀԱՇՎԱՐԿՆԵՐ ՀԻՄՆՎԱԾ ՆՈՐ ԱՆՏԵՂՅԱԿ ՓՈԽԱՆՑՄԱՆ ՀԱՂՈՐԴԱԿԱՐԳԻ ՎՐԱ

Ե.13.05 «Մաթեմատիկական մոդելավորում, թվային մեթոդներ և ծրագրերի համալիրներ» մասնագիտությամբ տեխնիկական գիտությունների թեկնածուի գիտական աստիճանի հայցման ատենախոսություն

Գիտական ղեկավար՝
ՀՀ ԳԱԱ ակադեմիկոս, տ.գ.դ.
Գ. Հ. Խաչատրյան

Երևան – 2016

INSTITUTE FOR INFORMATICS AND AUTOMATION PROBLEMS OF NAS RA

Davit Hakob Danoyan

SECURE MULTI-PARTY COMPUTATIONS BASED ON NOVEL OBLIVIOUS TRANSFER PROTOCOL

Thesis for obtaining candidate degree in technical sciences in specialty 05.13.05 “Mathematical modelling, numerical methods and software complexes”

Supervisor:
Academician of NAS RA, Doctor of Technical Sciences
G. H. Khachatryan

Yerevan - 2016

Table of Contents

Introduction

Secure computations were introduced by Yao in searching for a solution for the Millionaires problem, with two millionaires wanting to find out who is the richest of them without revealing to each other or any other party their actual possessions. In 1982 Yao published a two-party protocol known as garbled circuits' (GC) protocol for this problem, which still serves as the basic approach to the problem of secure two-party computation.

A generalized version of the above problem is called secure multiparty computation. Suppose mutually distrustful parties $P_1, \dots, P_n$ are willing to cooperate for computation of a function, which is preliminarily agreed upon, with their secret input data. The computation might not include any trusted parties and nothing should be revealed to any party or coalition of parties about someone's input beyond what could be implied from the function output itself.

First protocols solving secure multiparty computation problem were the GMW by Goldreich, MIcali and Widgerson and the CCD protocol by Chaum, Crépeau and Damgård in 1987 and 1988 respectively. Although being intended for the same problem, these protocols have many differences, among which are the security assumptions and function input method. Later several other protocols were suggested for the problem, some of which are described in the thesis.

Although the mentioned protocols and problems were of only a theoretical interest at time of their introduction, the advancements in information technologies in recent decades brings those into the field of practical interest. The first ever framework implementing secure computations was introduced in 2004 is known as Fairplay. It is based on Yao's GC protocol and proved that secure function evaluation (often referred as SFE) can be used in practice. Moreover, several enhanced implementations for GC protocol and also for

multiparty computations were developed in the following years. Implementations of GMW protocol were also proposed in different security settings, along with practical applications in online marketplaces [9]. Not to mention the implementation of SMC in real financial transactions for an auction held in Denmark in 2008. Other applications of SMC were also proposed for data mining, genome research and electronic voting.

Secure multiparty computation frameworks were introduced in the last decade and can be improved in implementation level and protocol design by applying novel software development and cryptographic techniques and technologies. As building blocks of those protocols are improved, they also can be integrated into available compatible frameworks for overall performance and security enhancement. Most of the protocols operate on Boolean or arithmetic circuit representations of functionalities. As underlying cryptographic primitives are usually used homomorphic encryption and oblivious transfer protocol. A novel approach based on white-box cryptography to the latter was proposed recently, which significantly enhances its performance. Integration of the protocol with other state-of the-art techniques into an oblivious transfer based SMC framework might also improve the overall performance.

The purpose of the thesis is to design and implement a generic framework for secure multiparty computation, construction of an extension protocol to white-box oblivious transfer protocol which will decrease the required invocation count of the latter, resulting in enhancement of the overall performance of the applications based on the framework and design of applications using the framework's capabilities. The lifecycle of implemented framework consists of the following stages:

- The participants of computation input a program implementing their desired function described in a high level programming language.

- A single pass Boolean circuit implementing the same function is being generated by the compiler module of the framework.
- The generated circuit is being executed securely based on GMW SMC protocol.

The subjects of this research are development of secure multiparty computation framework, oblivious transfer protocol extensions and privacy-preserving applications based on SMC framework.

The results are important for the following scenarios of practical interest:

- The framework consists of separate modules, which allows to integrate or replace those with new developments in the field, resulting in enhancement of overall performance.
- The framework can be used for implementing secure voting and rating systems, as well as for research in bioengineering.
- The proposed extension for GMW protocol adds computationally passive parties.

The results of the thesis have been presented and discussed in

- Regular seminars of the chair of Discrete Mathematics and Theoretical Informatics of Yerevan State University,
- 10th International Conference on Computer Science and Information Technologies CSIT-2015, (September 28 - October 2, 2015, Yerevan, Armenia),
- 10th Annual Scientific Conference of the Russian-Armenian (Slavonic) University, (November 30 - December 4, 2015, Yerevan, Armenia),
- Regular seminars in “Applied Cryptography Laboratory” of AUA.

The scientific novelty of the thesis includes design and implementation of a generic framework for secure multiparty computation, which:

- uses white-box cryptography based oblivious transfer instead of public-key methods in field of SMC;
- constructs an extension protocol to white-box oblivious transfer (WBOT) that reduces required WBOT invocation count;
- enables participation of computationally passive participants.

The implemented framework for secure multiparty computations is implanted by "SHANT COMPU" LTD within the software implementations for IPSC - Institute for Political and Sociological Consulting for development of anonymous rating system and is being tested at the point.

The main points presented for the defense are:

- An SMC framework using white-box cryptography techniques securely in semi-honest adversarial model.
- Justification of the performance enhancement compared to existing implementations.
- An extension for white-box oblivious transfer protocol.
- An extension for GMW SMC protocol allowing computationally passive participants.
- An application for combining competing services into unified ordering system securely.

List of publications

A. H. Jivanyan, G.H. Khachatryan, T. V. Sokhakyan and D. H. Danoyan. Acceleration of Secure Function Evaluation Protocol. Computer Science and Information Technologies (CSIT), 2015, pp 115-118.

D. H. Danoyan and T. V. Sokhakyan. A Generic Framework for Secure Computations. Proceedings of the Russian-Armenian (Slavonic) University #2, Physical-Mathematical Sciences, 2015, pp. 14-21.

D. H. Danoyan. Extending White-Box Cryptography Based Oblivious Transfer Protocol. Proceedings of the Yerevan State University #1 (239), Physical and Mathematical Sciences,2016, pp. 40-44.

D. H. Danoyan, "Secure Multiparty Computations for Collaboration Between Competing Service Providers", Transactions of IIAP of the NAS RA, Mathematical Problems of Computer Science, vol. 45, pp. 59-66, 2016.

Chapter 1. Background

This chapter includes basic definitions and notations used in the dissertation. Background information in the domain of secure computation and oblivious transfer protocols, along with introduction to cryptographic primitives used as building blocks in our and alternative frameworks or essential for their analysis.

1.1 Common Definitions

In this section we define and describe cryptographic primitives and terminology, which will be used throughout the thesis. Most of the definitions are common to the field and can be found in literature. For full background and additional information [28, 29, 70] can be used.

1.1.1 Security and randomness

Several security levels are defined for cryptosystems or algorithms security.

Schemes with information-theoretic security provide secureness which does not depend on the computing power of the attacker – thus the attacker cannot break the scheme even while possessing unlimited computing abilities, because the information available to him throughout the processing of the scheme is not sufficient for making conclusions about the encrypted data, so cryptoanalytic techniques do not help here.

In the special case called perfect security of the above defined security level, the result of the encryption algorithm doesn't provide any information on the original message text if the secret key is not available. In this case for each message m from the message space must exist a secret key k, such that $E(m,k) = c$.

Where E is the encrypting function of the perfectly secure scheme.

Another security level for practical purposes that is weaker than the above mentioned levels is called semantic security. A widely spread definition of the latter is provided below, as can be found in many sources:

A cryptosystem is semantically secure if any probabilistic, polynomial-time algorithm (PPTA) that is given the ciphertext of a certain message m (taken from any distribution of messages), and the message's length, cannot determine any partial information on the message with probability non-negligibly higher than all other PPTA's that only have access to the message length (and not the ciphertext) [49].

Although, semantic security doesn't guarantee secureness in information-theoretic model, it is considered enough for many applications, among which are many widely known encryption schemes, which are based on assumptions that some problems' solutions are computationally hard.

Suppose we have two random variables $A(k,x)$ and $B(k,x)$ where the $k \in K$ is a security parameter and $x \in X$ is an input parameter. Let $A = \{A(k,x)\}_{k\epsilon K, x\epsilon X}$ and $B = \{B(k,x)\}_{k\epsilon K, x\epsilon X}$ be the distribution ensembles for the variables. We say that A and B are *computationally indistinguishable* from each other, if for any non-uniform probabilistic polynomial time algorithm D the following is true:

$$|\Pr[D(A(k,x)) = 1]_{x\epsilon X} - \Pr[D(B(k,x)) = 1]_{x\epsilon X}| < \delta(k)$$

Where $\delta(k)$ is a function that can be considered negligible in k, which means that for every polynomial $p(k)$ exists a K, so that for every

$$k' > K, \delta(k) < {}^{1}/_{p(k)}$$

In other words A and B are that no probabilistic polynomial time algorithm (PPTA) can discriminate between them (with non-negligible probability).

Claude Shannon defined two basic properties (called confusion and diffusion) a cypher should possess to be considered secure.

The *Confusion* property is present, if encryption algorithm alters the plaintext in such a manner, that the ciphertext differs radically from it.

The *Diffusion* property stands for the fact, that alteration of a single character of the plaintext will result in many changes in the ciphertext. If no or weak diffusion is applied through the encryption process, the cryptanalysis of the scheme would be much easier, with using slightly changed plaintexts for finding out the initial plaintext for given ciphertext.

These are the definitions commonly used in literature that can be found in [28, 72, 73] and elsewhere.

1.1.2 Symmetric-key cryptography

Symmetric-key cryptography involves the encryption/decryption techniques where both the sender and the recipient have the same key (see Fig. 1). Also encryption schemes with not identical keys in possession of encrypting and decrypting parties are categorized as symmetric-key schemes if the keys are easily derivable from each other.

Symmetric-key scheme is a pair of algorithms used by parties - $E: K \times M \rightarrow C$ and $D: K \times C \rightarrow M$, where K is the key space, M is the message space and C is the ciphertext space. The sender uses the encryption algorithm E to encrypt the message $m \epsilon M$ using the key $k \epsilon K$ and transmits $c = E(k, m)$ to recipient. The recipient decrypts the ciphertext with the same key k and gets the initial message text: $m = D(k, c)$. To making the scheme practical, the secret key should be protected from the eavesdroppers. The parties should agree upon the key in advance or have a secure method of communication.

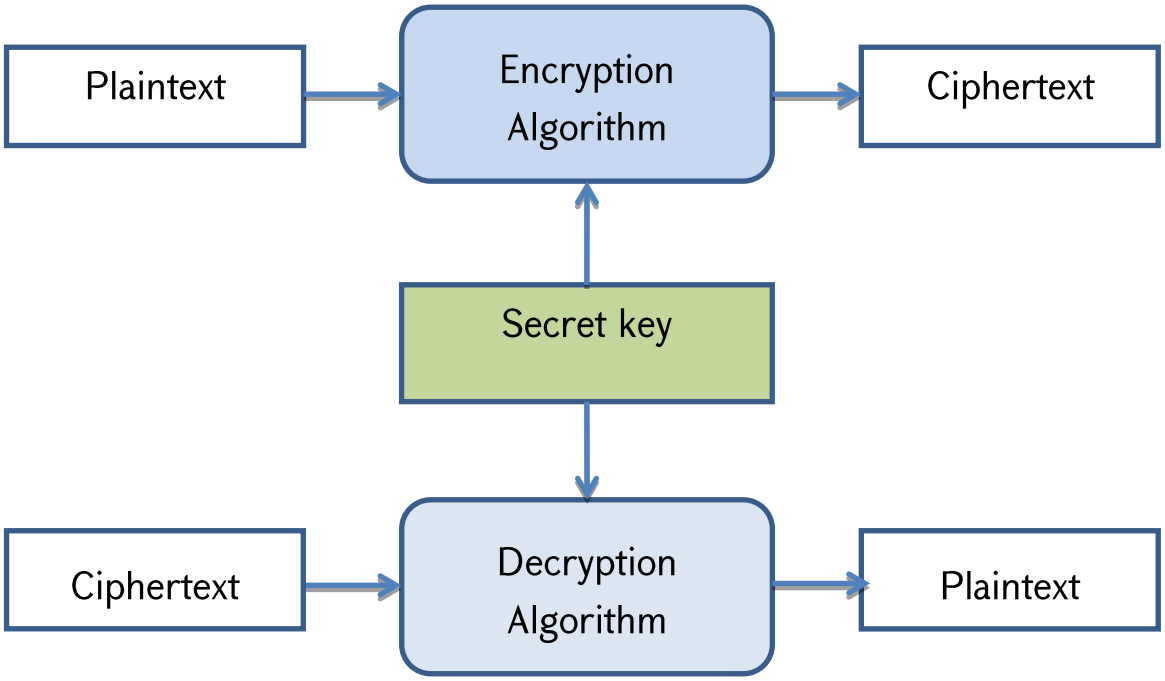

Fig. 1: Symmetric-key systems

Symmetric-key encryption schemes use either block or stream ciphers, both of which are widely used. Brief descriptions and use cases can be found below.

1.1.2.1 Block Ciphers

Definition: Block cipher with n-bit block and k-bit key is a pair of methods (E, D) where the following properties are satisfied.

$$E: \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n, D: \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$$

$$D\big(k, E(k,m)\big) = m, \qquad k\epsilon\{0,1\}^k, \qquad m\epsilon\{0,1\}^m$$

The key size is independent of the block size, while the plaintext and ciphertext are of equal length. As the latter two are from the same space, we can infer that the secret key is used by the encryption algorithm to make a permutation over the message space.

By definition a block cipher can be applied to encrypt messages of fixed length only. However, plaintexts required to encrypt are usually significantly longer than the key length. Extension of the scheme to handle longer plaintexts is

done by dividing them to block sized portions, which are processed one by one.

If the block size is not a multiplier of plaintext length, the last block will be a short one, which should be somehow extended to match the size rule. Just appending zeros in the end is not an acceptable solution, because it may result in ambiguous situation for the receiver while decrypting. There were developed several padding techniques to overcome this problem that were proved later to be vulnerable [2, 3]. Currently there is a standardized technique for padding secure against padding oracle attacks [2].

1.1.2.2 Substitution-Permutation Networks

For managing long plaintexts iterated block ciphers are used. The encryption is done in several rounds and each round is the application of the round function to one message block. The round function is invertible and in some schemes is fed with different keys in each round to strengthen security. The sequence of operations used is called a substitution-permutation network (SPN), a sketch of which can be found in Fig. 2.

Using the same key to encrypt all blocks will result in vulnerability, because in case of repeating blocks the deterministic encryption algorithm will output the same ciphertext. Having a significant amount of ciphertext the attacker can spot patterns and get advantage. To avoid this security leak several techniques have been developed to add randomness and result in different ciphertext for same message.

Block cipher operation modes specify the exact way of applying block cipher to message. The most trivial mode of operation is the Electronic Codebook mode. In this mode the message is divided to blocks of a defined size and encryptions is applied to each block independently.

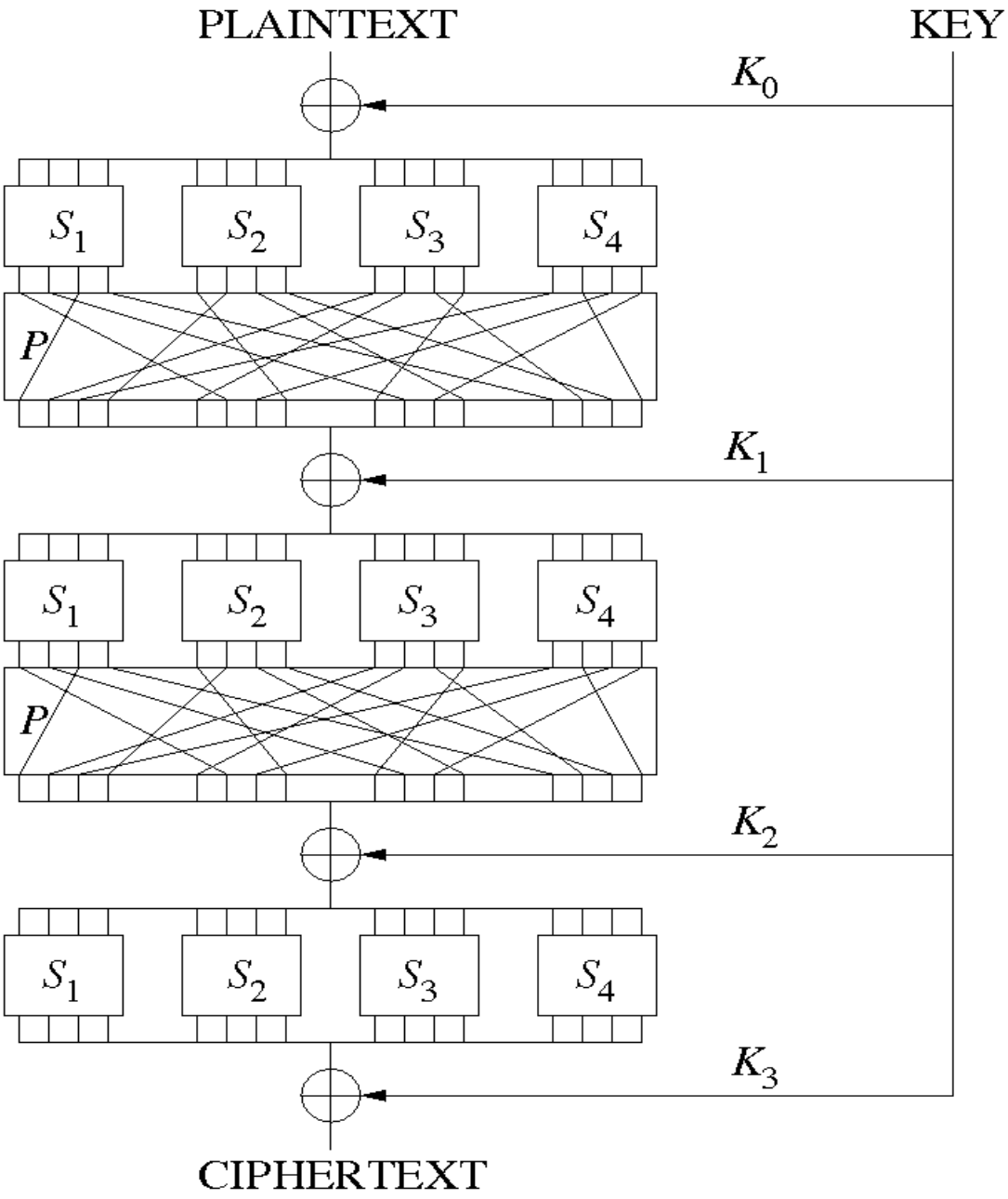

Fig. 2: Substitution-Permutation Networks [71]

Decryption is done the same way. The advantage of this mode is the independence of each step from the other one, which results in possibility to parallelize both encryption and decryption and save time. However the flip side of the coin is the feasible attack based on patterns in ciphertext corresponding patterns. The adversary can also gain advantage by correctly guessing the message organization. An example of this is attack based on header

information or other data expected to be at certain positions, when the attacker knows that the message is a file type.

1.1.2.3 Stream Ciphers

Another basic type of symmetric encryption is the so called stream cipher. As can be derived from the name, in this flavor the messagetext is not divided into blocks, but is rather encrypted in a "stream" bit by bit. A pseudo-random bit sequence is being generated using the symmetric key, which is thereafter used to alter the messagetext bits. In the primitive case each bit of messagetext is encrypted by XOR-ing it with appropriate bit of the generated keystream.

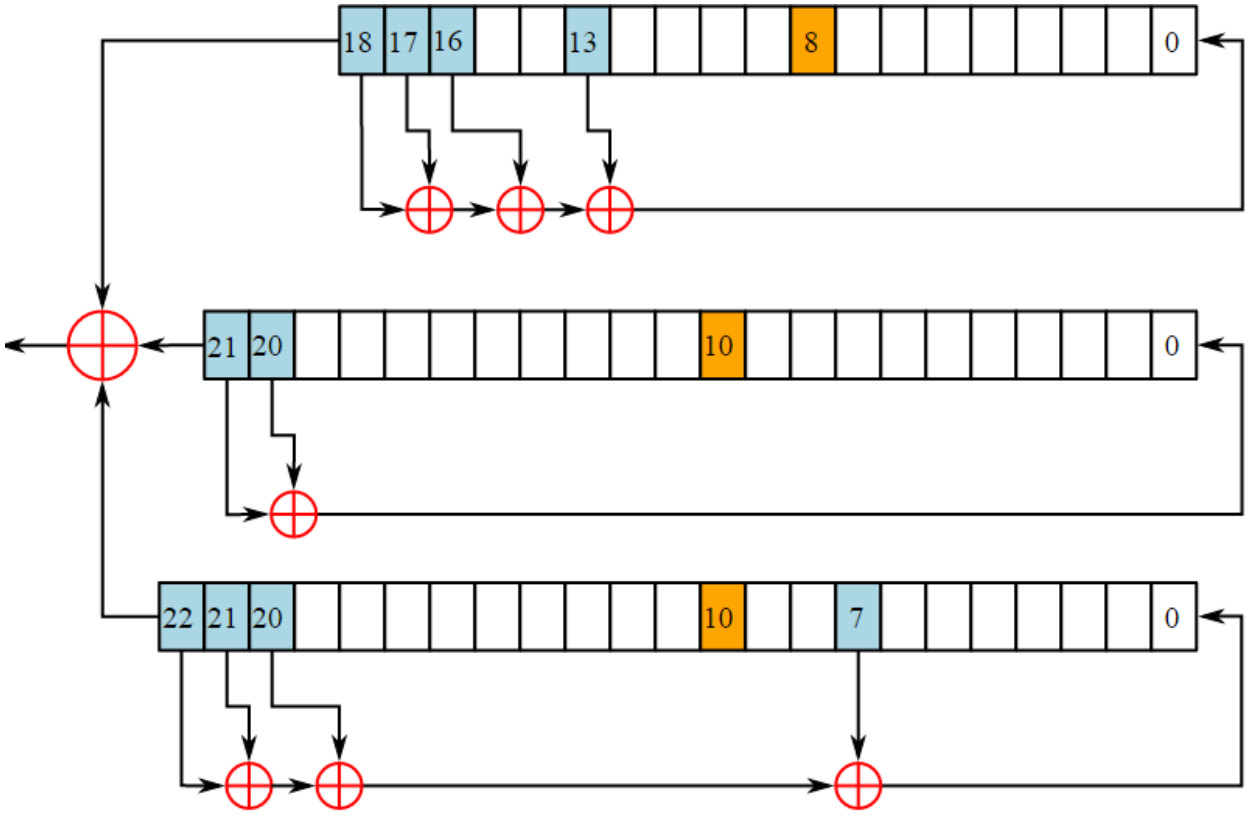

Fig. 3: Stream cipher [71]

It would be inappropriate to state, that stream ciphers can be considered as block ciphers where the block size is 1 bit, because here the keystream generation is carried out linear shift registers and the secret key is used as a seed for the pseudorandom generator. But this type of ciphers have an

obvious advantage against block ciphers regarding the processing of data shorter than the block cipher block length. In this case, the block cipher has to apply several techniques to avoid simple attacks, while stream ciphers work perfectly on this type of data.

1.1.3 Public-key cryptography

The common definition of public key cryptography, which can be found in literature is the following:

Public-key cryptography, or *asymmetric cryptography*, is any cryptographic system that uses two kinds of keys: public keys which are distributed among several parties (or publicly) and private keys which are only known to the owner [71].

Each key from the key-pair is intended for a different purpose – the public key is used to encrypt messages (so encryption can be done by any party possessing the public key) and the private key is used for decrypting the messages encrypted with the paired public key (so only the key owner can decrypt messages encrypted by the asymmetric encryption scheme) [71].

However, the strength and security of a public-key encryption scheme depend on the generation of private key, which are difficult to be derived from their public key due to computational impracticality. This nature of cryptography scheme delegates the problem of security to maintenance of the secrecy of the private key and distribution of the public key does not disrupt the security of the encryption scheme in any way.

The symmetric encryption is vulnerable, since the secret key should be agreed upon and the agreement process might be exposed to adversaries. In 1976 Diffie and Hellman published a new asymmetric-key cryptosystem which offered a practical method for symmetric key exchange without using a prior shared secret. This method of key exchange is based on exponentiation in a finite field and is known as Diffie-Hellman key exchange [26].

Asymmetric cryptographic schemes usually utilize cryptographic algorithms, security of which is based upon hardness assumptions of some mathematical problems, for which no efficient solutions are known. These include but are not limited to discrete logarithm calculation problem, integer factorization problem, etc.

As the computational cost of public-key based encryption is very high, the common use case of it is encryption of small data blocks. As private channels are not required for key exchange, this schemes are are typically used for symmetric encryption key exchange, i.e. the exchange is secure and is not compromised. The transferred symmetric keys can be utilized for the encryption of large amounts of data, which would take several orders of magnitude more computation time if it was encrypted with public key. [72].

In Fig. 4: Key generation key generation process is described. A large random number is usually used for the generation of two keys to be used as public and private keys for the asymmetric encryption scheme.

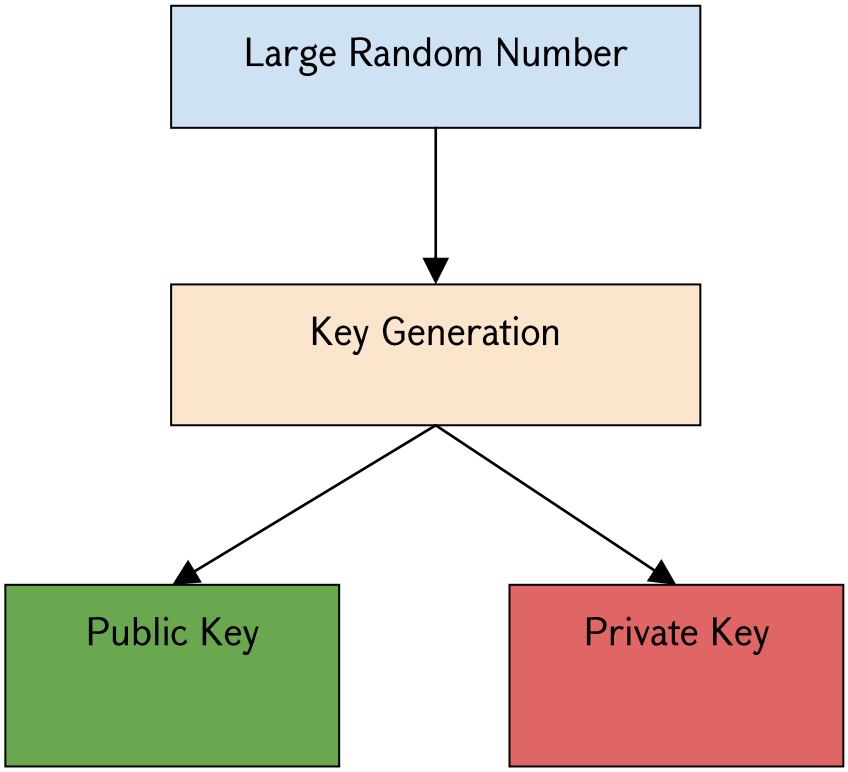

Fig. 4: Key generation

Diffie-Hellman key exchange is a specific method of securely exchanging cryptographic keys over a public channel and was one of the first public-key protocols[26]. The exchange process involves 2 parties, A and B. First, the parties agree upon integers g and p, where p is a large prime (typically at least 512 bits) and g is a primitive root modulo p. The numbers g and p might be shared with other parties. The parties A and B choose large random numbers a and b correspondingly as their private keys. A computes $A = g^a (mod\ p)$, which is being transmitted to B, while B computes $B = g^b (mod\ p)$, which is being transmitted to A.

Then, both parties compute key $K = g^{ab} (mod\ p)$ which is common for them. A finds K with the following formula:

$$K = B^a (mod\ p) = (g^b)^a (mod\ p)$$

While B uses:

$$K = A^b (mod\ p) = (g^a)^b (mod\ p)$$

The shared key K is available for A and B only it can be used for information exchange without other parties compromising it. In order to do so, adversaries should find $K = g^{ab} (mod\ p)$, with the knowledge of $g, p, A = g^a (mod\ p)$ and $B = g^b (mod\ p)$ only. This can be done by getting a from $A = g^a (mod\ p)$ and b from $B = g^b (mod\ p)$. So, the described attack comes to solution of the discrete logarithm problem, which is considered a hard problem for large primes. Computing the discrete logarithm of a number modulo p roughly takes as much computation time as the prime product factorization p, hardness of which is the underlying security assumption of the

RSA, thus providing security level to the Diffie-Hellman key exchange protocol similar to RSA cryptosystem.

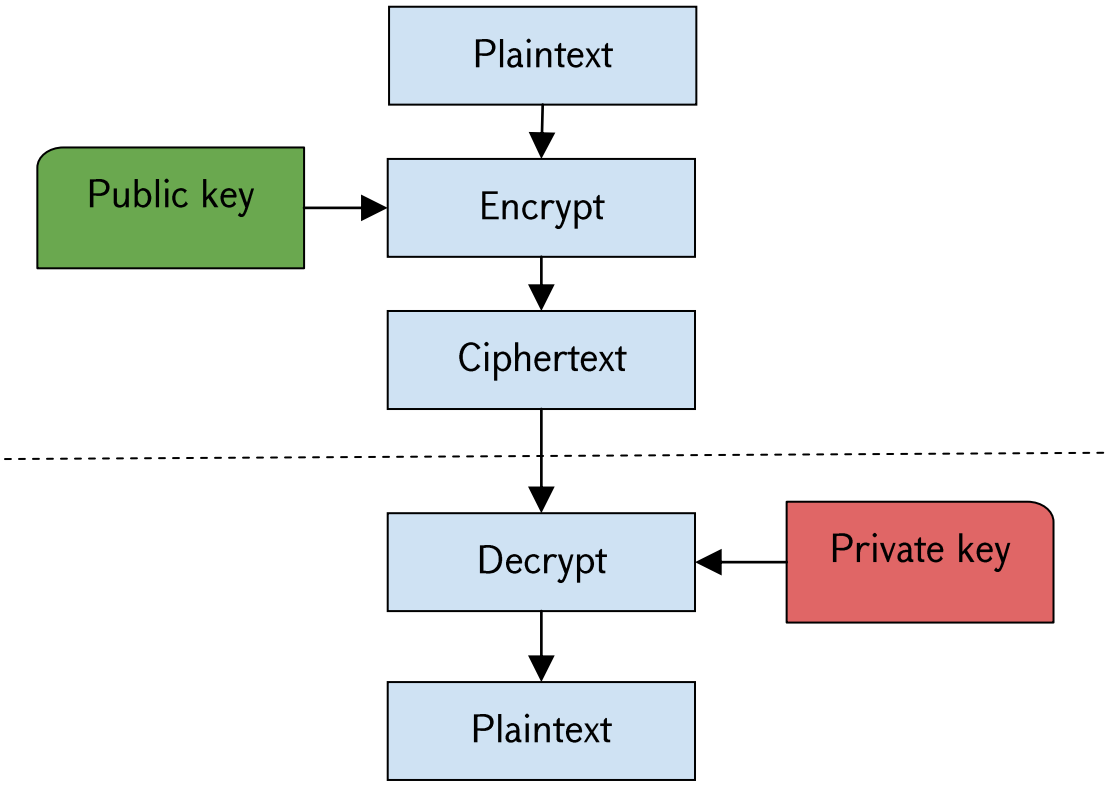

Fig. 5: Asymmetric key encryption scheme workflow

1.1.4 Homomorphic encryption

The class of encryption schemes that highly depend or are based on the principles of homomorphism is called homomorphic encryption. Where the term homomorphism is the mapping of two algebraic structures, where certain operations remain valid. To demonstrate this, we will consider the messagespace where a certain operation $\odot$ is defined between the plaintexts and another operation $\circledcirc$, which is defined between the ciphertexts. We will call an encryption scheme $(Enc_{key}; Dec_{key})$, where Enc_{key} is the encryption method and Dec_{key} is the decryption method, homomorphic, if for any messages m_0 ,m_1 from messagespace the below formula is correct:

$$Dec\big(Enc(m_0) \circledcirc Enc(m_1)\big) = m_0 \odot m_1.$$

A flavor of homomorphic encryption is called additively homomorphic because of homomorphism is satisfied over the addition operation, so the $\odot$ is addition operation in messagespace and $\circledcirc$ is the addition operation over the ciphertexts. However, if the ciphertexts can be processed in a limited way only (not every operation can be carried out over the ciphertexts that will result in encryption of the plaintext derived from the same operations applied to appropriate plaintexts or vice-versa) the cryptosystem is called partially homomorphic encryption scheme, otherwise, if any operation carries homomorphic nature, the encryption system is called fully homomorphic. Known encryption schemes based on partial homomorphism include but are not limited to the original RSA [17] and ElGamal [27]. There have been developed specific protocols for secure computation of some functions based on partially Homomorphic cryptosystems [82 - 84]. Anyway, the limited number of operations that can be carried out over the ciphertexts, greatly limits the way of interaction and creates obstacles for carrying out secure multiparty computations. On the other hand, fully homomorphic cryptosystems [85], which can support for example operations of multiplication and addition, allows evaluation of circuits with any depth and keeping the size of the cyphertext independent from the generating function complexity, can be useful for generic function evaluation. These three properties were suggested by Rivest et al. in [85] and the first prototype of it was developed in [86] by founding the scheme on somewhat homomophic cryptosystem, where the operations are supported, but evaluation of circuits is limited, in terms of the cicuit depth. With use of such a scheme Gentry applied the so called bootstrapping technique, because while the maximum depth of the underlying somewhat homomorphic cryptosystem exceeds the scheme's decrypting operation circuit, the homomorphically decrypting operation can be applied on

the ciphertext, resulting in a new ciphertext, which can be processed in more ways. This technique is explained in detail in the original paper of Gentry, which is referenced above.

The idea of somewhat homomorphic encryption was later utilized in several other cryptosystems [87, 88]. This cryptosystems are able to process circuits with any depth, however the maximum depth should be known when the secret key is being chosen.

Fully homomorphic encryption schemes have usage not only in computations carried out on encrypted input, but also in other applications as reusable garbled circuits [90], verifiable computing [91], and homomorphic signatures [92]. However, the fully homomorphic encryption schemes available now are too impractical for utilization in secure computations, but some protocols are being developed out of theoretical interest, because this type of schemes are able to provide information-theoretic security for the secure computations.

1.2 Secure computations

Overview

Secure computation protocol is a process specified two or more mutually distrustful participants who want to cooperate upon computing a function with their inputs, while keeping their inputs secret from each other and any other entity. In this section we provide security definitions, adversarial models and also introduction to two types of secure computations – secure two-party computation and secure multiparty computation, with implementing protocol examples of those.

Participants of secure computation expect to emulate through protocols of secure computations a trusted party, to whom they would send their secret inputs without being concerned about the security and that trusted party would send them the result of computations, without revealing secret at any stage of the overall process. So, parties not concerned with the implementation details

of protocol can treat it as a virtual trusted party. Security and correctness of protocols are often backed with putting their operation into comparison with the same functionality implemented with presence of a trusted party. This approach is referred to as real/ideal model paradigm (Fig. 6).

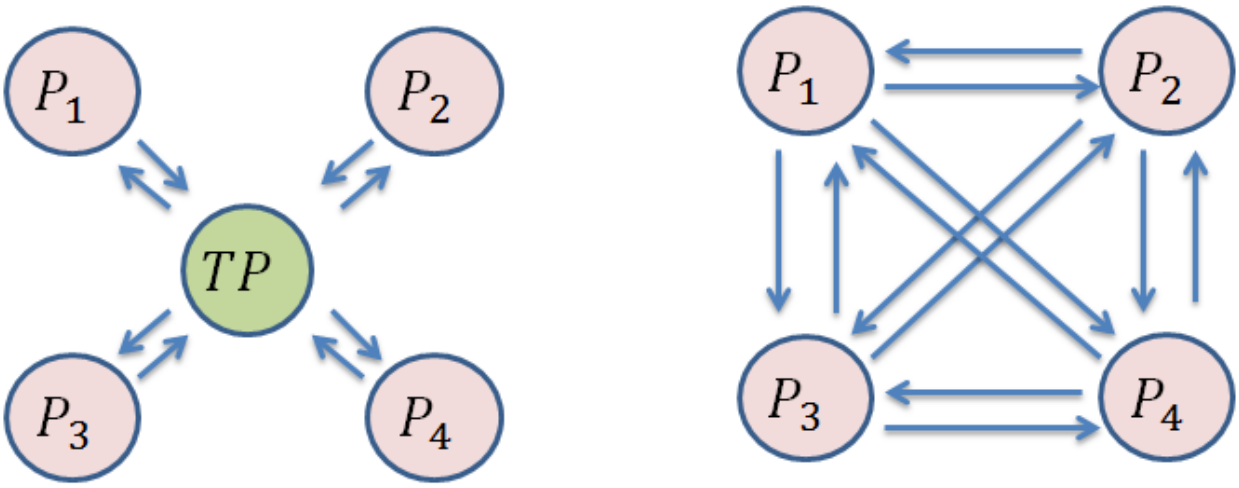

Fig. 6: Real/ideal models

First secure computation protocol was introduced by Yao in [30] for the two-party setting and later came out several protocols for the multiparty setting [1, 5]. We will discuss the specifics of two-party and multiparty settings after defining security models.

1.2.1 Security models

Adversary is an entity, who tries to attack a secure computation protocol execution. An adversary might have various motivations and purposes for attacking SMC protocols, such as gaining knowledge of private data, altering protocol output for one or more participants or even abort the protocol execution. A participant is called *corrupted*, if it is controlled by an adversary.

There are several properties that a protocol should hold for being secure against an adversary. However, a protocol can be secure against one type of

an adversary, but insecure against another. Different adversarial models will be discussed later in this section. For now, we present some properties a protocol is required to demonstrate to be interpreted secure against a fixed adversary.

- Secure computation protocol should keep *privacy*: None of the participants should obtain any more information, than is specified by protocol and/or can be inferred directly from her input and protocol output. An example, of learning information about other participants output without cheating without compromising protocol's security is deducing that all participants had bits lower, than the winning bid's value in a protocol implementing auction.
- Secure computation protocol should guarantee *correctness of output*: All parties involved in protocol execution should be sure of the result he got is not questionable. For the example of auction, every participant is guaranteed that the outputted winning bid is the highest of inputs.
- Protocol also should guarantee *independence of inputs*: Inputs of corrupted participants should not depend on other parties' inputs.
- Non-corrupted participants should be guaranteed to obtain their respective outputs after the execution of the protocol, so the corrupted ones should not be able to abort the protocol (also this property is omitted for several protocols, where the execution is being aborted to prevent the adversary to access private information).

In the following sections we will define and discuss the usual adversarial models applied to secure computations.

1.2.1.1 Semi-honest model

A protocol is called secure in semi-honest security model (or secure against semi-honest adversary model) if participation of semi-honest adversaries in the protocol execution does not compromise the security of the other participants.

A semi-honest adversary (also often referred as honest-but-curios) informally speaking, in a secure computation protocol a party demonstrating semi-honest behavior follows the protocol properly, although it tries to cheat and get additional information by means not restricted by the protocol. This means, that he sends messages, executes required functionality, but records his computational activity and also all communication he was involved in. In other words, the semi-honest adversary is honest with regards to following the protocol, but not "fully" honest, because she does not miss a chance to gain additional information not present in the protocols output.

Also there are several properties that can be incorrectly attributed to semi-honest model, because of wrong interpretation of its definition. Accordingly, secure computation protocols should be designed in a way to avoid such assumptions inconsistent with security.

Collusions are not precluded in semihonest model

While considering semi-honest adversarial model, we assume that all parties strictly follow the protocol. The adversaries in this model can try to get additional information from the communication without breaking the protocol. Actually this definition also doesn't exclude the possibility of two or more parties being in agreement or even controlled from one source. Here we bring an example of a protocol secure against not colluding adversaries but leaks data when two parties cooperate.

Consider n parties, each one with a single integer input x_i $0 \leq i < n$, desiring to compute the sum of their inputs $x_0 + \ldots + x_{n-1}$. First party chooses a random value r and passes $y_0 = x_0 + r$ to second party. Each

party P_i gets some value $y_{i-1}$from P_{i-1}, which is in fact $r + x_0 + \cdots + x_{i-1}$, add its own input x_i to the current value and passes it to the next party. The last party P_{n-1} adds its input to the value he got from previous party and passes it to P_0. P_0, the only one knowing the value r, gets the sum by subtracting r from y_{n-1} and sends it to all other participants. Obviously this protocol is secure when all parties are isolated.

However, when two parties, that are not aligned consecutively, are in cooperation with each other, they can get additional information on inputs of other participants. Consider P_i and P_j $i < j - 1$ are under control of an adversary. During the protocol evaluation gets two current values y_{i-1} and y_{j-1}. Using this advantage he can calculate $y_{j-1} - y_{i-1} = x_{i+1} + \ldots + x_{j-1}$. This is obviously a security leak and therefore the above protocol is not secure against adversaries with semi-honest behavior.

Protocol workflow depends on the input

Many functions can be implemented with use of several different algorithms and the choice of the latter often is based on efficient use of execution time, memory, or network communication However, in case of secure computation some of the very efficient approaches may be sacrificed to gain security. Here we bring one of the frequently referred examples showing that the protocol workflow depends on the input.

For the sake of simplicity, the example is provided for the two party case. Consider participants desiring to find out if their secret messages of the same length are equal. A trivial method for solving this problem can bethe execution of bitwise comparison (compare appropriate bits of the parties' inputs from the most to the least significant), execution of which is stopped with negative answer, if in any step non-equal bits are found and a positive

answer is outputted, after all appropriate bits have been compared and been equal. This means that if the most significant bits of the secret messages differ, the algorithm will cease after only one comparison operation executed and only l comparison operations will be executed, if the messages differ in the l-th position.

So if messages differ in the l-th position, the executing party can obviously deduce, that the first $l-1$ bits of the secret messages match, however, this information cannot be gained, while the computation is done in the so called ideal execution model, where the inputting participants provide secret messages and the learn the results only. The guilt of the information leak is definitely the heavy dependence of the algorithm flow on the secret messages being compared.

Information leaks with deterministic encryption

Data encryption or hash function application on it does not always provide guarantees for privacy. The encryption or hashing method chosen may cause data leak during some computations. Below we bring a simple private set intersection algorithm, which shows the mentioned vulnerability appears to be real.

Two parties wish to find out the common elements of their private sets containing secret messages in their possession, which are $\{x_1, \dots, x_l\}$ for the first party and $\{y_1, \dots, y_l\}$ for the second one. A non-colliding hash function which also appears also deterministic can be viewed as a secure method to be applied for hiding the items. So, the parties encrypt all of the elements in their private sets with the use of the mentioned hash function and put the resulting encrypted list in a shared location, where each of them searches the elements of the opponents encrypted set in his own and if an element is found in both encrypted sets, than each party looks up the original item in the initial set and

marks it as common. The intersection of the private sets can be found in this way, but there are some methods of finding out more information, than the protocol intends to give. If the messagespace of the sets elements is appears to be of a small size, each of the parties can apply the hash function on all messages in the space and after the encrypted set of the opponent appears in their shared location, he can easily deduce all original elements of the latters' set.

Even in case of a huge messagespace, information leak is also possible. In this case the attacking party can search in the set of the opponent for select items he is interested in.

Both of the leak cases provided above are the result of using a deterministic function for encrypting secret messages, which results in information leak, which is impossible in the ideal execution model, where the parties are provided with input and output data only. For avoiding such situations, encrypted items that should at some point be shared with other parties should be encrypted with semantic security guaranteed encryption schemes.

1.2.1.2 Malicious model

The security of protocols should be considered in so called malicious adversarial model, if the participants of the protocol desire to gain information from the other parties even at the cost of deviating from the protocol. Sometimes this security model is referred as active adversarial model also. There are no any rules regulating the behavior of a malicious participant, so he can deviate from the steps of protocol in arbitrary manner, which can result in different problems as information leak, wrong output of protocol or ceased execution of protocol.

A malicious attack known as "selective failure attack" is comprehensively discussed in the work of Mohassel and Franklin presented in [75]. The attack is used against the evaluator in the Yao's garbled circuits secure two-party

computation protocol by a malicious circuit generating party, which sends wrong wire labels to the evaluator during the execution. After the evaluation is completed, the malicious party finds out, if a preliminarily determined input bit of the evaluator is 0 or 1, depending on successfulness of the evaluation.

1.2.1.3 Covert adversaries

The protocols designed to be secure against malicious adversaries are very slow in comparison with the ones secure in the semi-honest model. But the latter protocols are not applicable for many real life scenarios, where parties of the secure computation are not guaranteed to strictly follow the protocol. So a security model placed in the middle of passive (semi-honest) and active (malicious) adversarial models both in terms of security and effectiveness of the protocols was introduced to cover the cases, where the attacker may deviate from the protocol, but the deviations are somehow limited. This security model often referred as covert adversarial model introduced and described in details in [76, 77]. In this model, a cheating participant is captured with some probability, but still can learn additional information about the opponent's inputs.

Aumann and Lindell [76] constructed an example of two-party protocol with covert security the performance of which is a small factor less compared with baseline Yao's garbled circuits protocol. For further details we refer the reader to original publication of Auman and Lindell.

1.2.2 Oblivious transfer

The oblivious transfer (OT) protocols is extensively used during secure computations. The initial idea of OT is introduced by Rabin [18]. This protocol involves two parties the sender and the receiver. According the original description, the sender sends a message to the receiver and remains oblivious whether it got to the receiver or not, while the receiver gets the message with

probability $\frac{1}{2}$. The modern form of OT protocol being widely used and known as 1-out-of-2 oblivious transfer (OT_2^1) is introduced by Even et al. [12]. As previous form, it involves two parties the sender and the receiver. But here the sender has two indexed secret elements x_0 and x_1 having l bit length, and the receiver has a secret selection bit r. At the end of protocol execution the receiver holds the element x_r being unaware of $x_{1\oplus r}$ without leaking anything about selection bit r. A generalized form of OT_2^1 protocol is 1-out-of-n oblivious transfer (OT_n^1). Here instead of operating on two elements and selection bit the sender and the receiver hold n indexed secret elements and secret selection index $0 \leq i < n$. Formally OT_n^1 can be expressed in following $OT_n^1((x_0, \dots, x_{n-1}), i) = (\lambda, x_i)$. Namely the sender (first participant) gets no output, and the receiver gets x_i only.

Another generalization of OT protocol is $m \times OT_n^1$. Here the sender has database of m pairs of l bit strings $(x_i^0, x_i^1), 0 \leq i < m$, and the receiver holds vector of selection indexes $(r_0, \dots, r_{m-1})$. At the end of protocol execution the sender has no output, and the receiver has elements $x_i^{r_i}$, $0 \leq i < m$.

1.2.2.1 OT extension

The functionality $m \times OT_n^1$ described in previous section is equivalent of m virtual executions of OT_n^1 protocol. In 1996 Beaver introduced OT *extension* [74]. This technique works by replacing m direct executions of OT_n^1 protocol by small number of base executions allowing perform the rest of data exchange using only symmetric cryptography primitives which are order of magnate faster compared with public key (PK) operations. Essentially it reduces number of performed PK operations from $O(m)$ to $O(k)$, where k is the security parameter. Conceptually, this method is similar to usage of PK cryptography, when instead of processing a large message using expensive PK operations a

hybrid encryption scheme is used where PK operations are performed only for symmetric key exchange, and the later key is used to process large message efficiently.

Until 2003 this method has only theoretical interest as the original construction is not much efficient to satisfy real world needs. A better implementation of OT extension is proposed by Ishai et al. [78], requiring only three hash function invocations for each extended OT execution in semi-honest model (PK operations are used during base OT executions only). Further improved implementation is provided by Asharov et al. [79, 80].

Providing OT protocol security in malicious model has an expensive overhead compared with semi-honest implementation. For example OT extension secure in malicious model, provided by Nielsen et al. [23] is almost two times slower compared with original implementation which is secure only in semi-honest model.

1.2.2.2 Private information retrieval

Private information retrieval (PIR) is a cryptographic protocol intended for a client to choose an item to be obtained from a database without revealing information about the item he requested. PIR was introduced by in [15] and [23] using a single database for information-theoretic and computational security respectively.

In terms of requirements private information retrieval is a lighter version of $1 - out - of - n$ OT, because here the database is not secured against client with respect to restriction of getting information about other items. Implementations of this cryptographic primitive can be found in [18, 73].

1.2.3 Function representations

Secure computation protocols operate on a specific representation of the function, which is desired to be computed. When the desired functionality is already agreed upon, the participants of the secure computation need it to be described in a generic way. The most widely spread methods for representing a function are:

- Generating a Boolean circuit,
- Generating an arithmetic circuit.

Several secure computation protocols exist utilizing this representations.

1.2.3.1 Boolean Circuits

Boolean circuit is a very popular representation for a function, when it needs to be computed securely. Several protocols, including the so called GMW for multiparty computation and Garbled circuits for two-party computations are based on this representation of function. The formal definition, which is common in literature follows:

Definition: An n-arity gate g^n is a Boolean function which maps $\{0,1\}^n$ input bits to a bit called an output, namely, $g^n: \{0,1\}^n \rightarrow \{0,1\}$ [68].

The Boolean circuits are constructed using a universal set of binary operations, like XOR, AND and OR.

Definition: A Boolean circuit having n inputs, m outputs and v internal gates is a directed acyclic graph (V, E) with $|V| = m + n + v$ nodes and $|E|$ edges. Each node corresponds to an internal gate, an input or an output. Two nodes are connected with an edge if and only if there is a wire connecting corresponding gates. [68].

1.2.3.2 Arithmetic circuit

The other widespread function representation method is using arithmetic circuits, where the directed acyclic graph of the appropriate function is

consisted of nodes, which are operation over a ring $< R, \times, + >$. Nodes with no incoming edges are the input gates, and those who don't have outgoing edges are respectively the output gates of the circuit. The advantage of arithmetic circuits over Boolean circuits is usually the circuit size, which increases dramatically, if the function operates on a very large ring.

1.2.4 Secure two-party computation

As the world is getting more and more connected in many real world scenarios, parties with different and potentially conflicting interests have to interact. Examples of this are citizens and governments (electronic passport and electronic id), patients and health insurers or medical institutions (electronic health card, e-health services), or companies and service providers (cloud computing). In the mentioned context questions of paramount importance are the architecture and the ability of an underlying communication system to fulfil varied security and privacy requirements of the involved parties.

Protocols for secure computations enable two or more mutually distrustful parties to communicate and evaluate some commonly agreed functionality with private inputs from all participants. Yao pointed out that secure two-party computation protocols can be utilized for computation of arbitrary computable function [30].

Yao's protocol remains one of the most actively studied methods for secure computations. Although Yao never published a precise protocol, the very first real world implementation of secure two-party computation [37] used Yao's basic garbled circuit approach, and it remains the primary paradigm for the plenty of 2PC implementations that have been developed during the past eleven years [59 - 61].

Yao's protocol is very efficient in terms of two-party computations. Many applications are known, where it's essential to keep the data inputted by the participants private.

Efficient secure two party computation methods give a chance for collaboration to distrustful parties. Several applications were modelled based on two-party computation protocols. Biometric identification [62] and genome research, as well as analysis carried out over private databases are developed also, along with applications to online auctions [36], contract signing [12], etc.

1.2.4.1 Yao's garbled circuits protocol

Yao's garbled circuit protocol allows two mutually distrustful parties with inputs x and y to evaluate any computable function $f(x, y)$ on their input values without giving away any side information on the secret inputs, except what can be inferred from the output itself.

The main idea is that one of the participants (called *generator*) generates an scrambled version of the Boolean circuit C computing the function f, and the second party (called *evaluator*) evaluates the circuit. Note that reverse engineering techniques are not applicable to garbled circuit, thus the *evaluator* does not learn any intermediate value.

Suppose the *generator* has a Boolean circuit C with 2 fan-in gates computing the function f. At the first step the *generator* fixes some integer k and assigns two random looking bit strings w^0 and w^1 to each wire of circuit C (label w^b conceptually encodes value $b \in \{0, 1\}$ for the wire w). Then for the gate g having output wire w_k and input wires w_i, w_j *generator* prepares garbled table with following entries:

$$Enc_{w_i^{b_i}, w_j^{b_j}} \left(w_k^{g(b_j, b_j)}\right)$$

where Enc is an encryption scheme fixed by *generator*. The collection of all garbled gates is called garbled circuit.

1.2.4.2 Generator

Then the *generator* passes garbled circuit and mapping for the labels for the output wires to the *evaluator*. An interesting feature of garbled circuit protocol is that the *generator* is aware all associations between input values on input wires, so and it is able to transmit the garbled circuit construction to the other party. To obtain wire labels for his input the *evaluator* runs OT protocol described afterwards with the *generator*.

1.2.4.3 Evaluator

The evaluation of garbled circuit is done in a hierarchical way. Given labels w_i and w_j of input wires of garbled gate g the *evaluator* decrypts the appropriate entry of garbled table using keys w_i and w_j. When labels of all output wires are computed the *evaluator* sends the function output value to the *generator* using provided mapping for output wires.

1.3 White-box cryptography

1.3.1 Overview

In this section we first bring several attack contexts and analyze the differences, after which we concentrate on the white-box attack model and show several vulnerabilities that were among those that motivated the

introduction of white-box cryptography. Those include but are not limited to entropy attack and key-whitening attack, which are briefly described below.

White-box implementations of symmetric ciphers are motivated by the fact that nowadays vulnerability of an encryption scheme does often originate from the algorithmic construction of the latter, but the environment, where the scheme is operating on. Considering the diversity of computing devices and the operating systems of those, guarantees of general security for users cannot be assumed. But encryption/decryption functionalities are currently executed in those various environments in a regular manner for logging in to different services, private communications, financial transactions and more. Also, much information is being kept on remote servers which can also be considered as untrusted in the user's perspective. So existence of methods which enable the user to utilize secure operations in untrusted environments is a very relevant problem.

White-Box cryptography was introduced by Chow et al. in [66], where white-box version of the AES encryption scheme was constructed. The term white-box is used to emphasize, that no secrets appear in the implementation and the encryption process is available to be viewed by the attacker thoroughly. In the references paper several methods for overcoming vulnerabilities in white-box attack model were presented. In another work of the same authors a white-box implementation of DES cipher was also described [66].

While in this thesis we use white-box cryptography in the context of oblivious transfer and secure multiparty computations, there are traditional fields of applications for it also. Those include software executions on remote servers or cloud services and cryptographic methods utilized on smart-cards [63, 64].

For the construction of secure multiparty computation protocols we use a white-box implementation of the block cipher SAFER+ [33], but any other white-box encryption scheme can be used also.

Another widely spread application field of white-box cryptography is digital rights management (DRM).

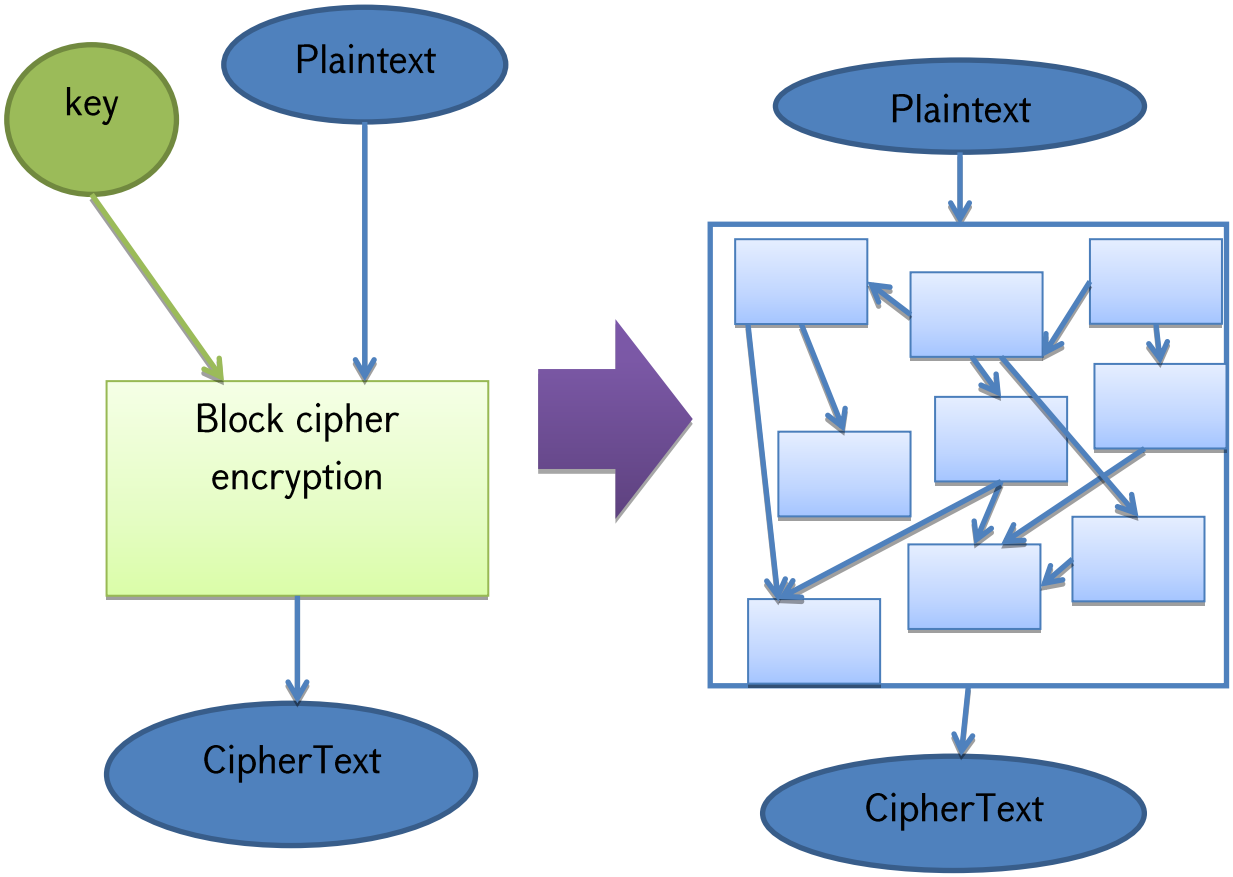

Fig. 7: White-Box implementation of a block cypher

The security problems with DRM applications are, in contrast to the others is that the untrusted party is the end user, and the content provider is considered as the party seeking security. Malicious clients try to extract the encryption keys used for protecting the paid content after they buy it (common examples of such are TV-shows, computer games and electronic books, which are downloaded by the user fully or partially encrypted and are decrypted locally after user purchases license) and later redistribute it or sell to other parties illegally, resulting in financial losses for the service/content provider.

In the black-box attack model the attacker has visibility of input/output data of the software. In the white-box model, on the contrary, the attacker has full visibility of the software execution, because the software operates in an environment fully controlled by him. This gives the attacker to observe the

intermediate results of the software execution and memory read/write operations by using debugging or disassembling tools. In the process the attacker can recognize the decryption operation and find out the secret key.
In 2004 Billet, Gilbert and Ech-Chatbi showed that the secret key hidden in the original white-box implementation of AES can be extracted in 230 work steps of their algebraic cryptanalysis [67]. This paper also inspired several other white-box implementations of symmetric ciphers to be constructed resistant to this and other attacks presented later.

1.3.2 Attacks in white-box context

Several attack techniques on symmetric-key encryption schemes are known that can be prevented with use of white-box cryptography constructions, although other prevention strategies are also available and described. Firstly, the entropy attack [45] by Shamir and van Someren is presented, after which another attack is presented known as key-whitening attack [44] by Kerins and Kursawe. The last attack we consider is a more recent technique based on acoustic noise produced by computing devices published in [48].

Entropy attack

Ideally, an encryption scheme should enable two parties to communicate and transmit secret messages over untrusted environments such as public networks, shared memory or untrusted computing devices and be assured that no one is able to intervene to the communication or steal the secret information. There are great advancements in symmetric-key cryptographic schemes that along with the enhancement of computing powers take little effort and time to secure the communication.
The main requirement for the execution of symmetric-key operations is that both communicating parties have agreed upon a secret key, which should be used for encryption and decryption. Obviously, the secret key should not be

easily guessed by the attacker, so it should be chosen randomly from key space.

Randomness of a key is measured by entropy [43]. The higher is entropy, the more reliable is the secret key.

Fig. 8: Memory dump graphical representation [45]

Shamir and van Someren invented an attack [45] on symmetric encryption schemes using the fact of the secret key having high entropy. They chose the way of seeking the hidden secret key in the memory of the program, instead of trying to compute it.

It turns out, that in "neighborhood" of such structured data as binary representations of compiled code and other non-secret information the secret key with high entropy is easily recognized. This approach can be applied, in situations, where the attacker has access to random access memory and/or hard disk.

Key whitening attack

This attack technique is applied to a narrower range of symmetric ciphers, which is the block ciphers that use key whitening. As described in previous sections, block ciphers usually have several rounds of operation and the key whitening technique, which is generating a new round key from the master key for every round, is important for defense against exhaustive search or other attacks. Kerins and Kursawe constructed an easy attack [44] on block cipher implementations, where the key whitening technique is applied.

The attack is not applicable to all such ciphers, because another requirement for vulnerability against this attack is that the S-Boxes of the cipher should static. As the S-Boxes of all major substitution permutation network block ciphers are publicly known, an attacker can easily find those in the program binaries. As the attacker may have the capability of also modifying the memory, he can substitute the original S-boxes with so, that she would get advantage.

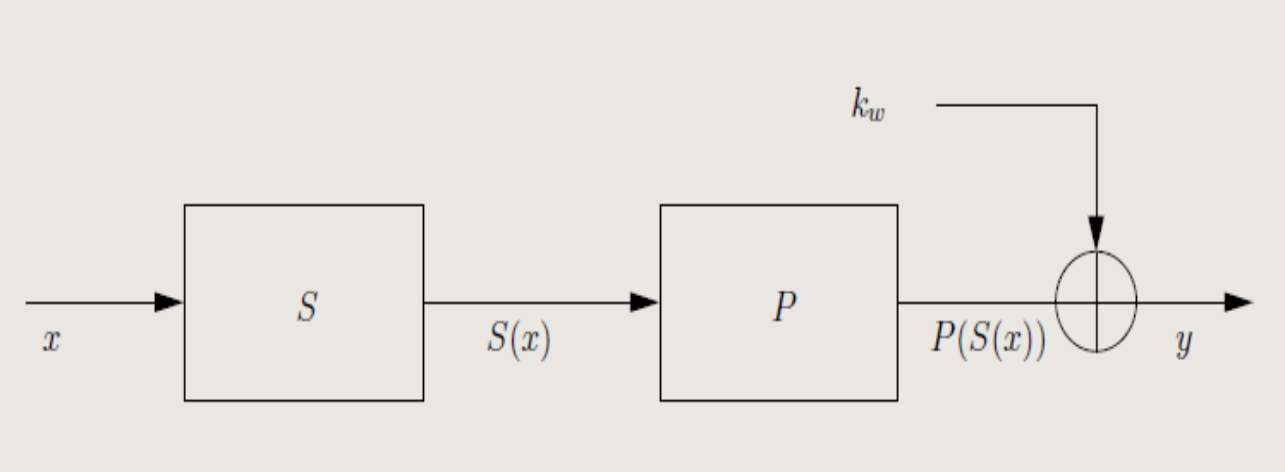

Fig. 9: SPN round

If the S-Box is replaced with zero matrix, S(x) would be a zero vector and any permutation of it would also be zero, so $k_w \oplus P\big(S(x)\big) = 0$, which means the hacked program will output the key, resulting in a successful attack.

This attack can be prevented with the use of block ciphers, where the S-boxes are key dependent, such as Blowfish [46]. Another approach is to use various verification methods for protecting the program from being modified.

Chapter 2. Extending White-Box Oblivious Transfer

Overview

In this chapter we discuss the cryptographic primitive called Oblivious Transfer (OT). We bring some historical information about the generation of the concept and its different flavours. Also description and verification of most basic OT protocol variants are presented accompanied by efficiency analysis and optimization/modification techniques. The first paragraph is dedicated to OT protocol by Rabin with its development into $1-out-of-2$ OT. Several schemes are given along with discussion of their relationships. Second paragraph is about $1-out-of-n$ oblivious transfer. Various construction techniques are brought accompanied by deliberation of performance and efficiency issues. In the third paragraph we present a novel approach to the concept of OT introduced in [20] based on White-Box Cryptography and consider its efficiency and security in comparison with the traditional techniques. The final fourth paragraph we dedicate to oblivious transfer extensions. We bring the motivation of applying those and present Beavers seminal approach to the problem. Other extension techniques are also discussed and our novelty of constructing extension for white-box cryptography based OT protocol [52] is presented, along with its correctness and security proofs.

2.1 Oblivious transfer protocols

The oblivious transfer as a cryptographic primitive was first introduced by Rabin in [10], which is considered as the seminal paper on oblivious transfer. A very similar idea laid behind [16] which was independently developed in early seventies. Wiesner's paper, which was first rejected by IEEE ITS and after

being published a decade later it introduced the quantum cryptography. The suggested primitive was called conjugate coding which was encoding two messages in a way that only one of them could eventually be retrieved after transmission. Rabin, on the other hand, designed a protocol for exchanging secrets between two parties asynchronously (not using simultaneous exchange techniques) and without participation of trusted parties. While implementing this protocol he brought another two party protocol, in which first party transmits information to the second and stays completely unaware whether the she received the information. Later another protocol was developed related protocol was proposed by Even, Goldreich and Lempel called $1 - out - of - 2$ oblivious transfer, which currently is usually referred as OT. Although the latter differs from Rabin's OT in terms of functionality, these two are equivalent (proved in [13]). Further improvements in terms of performance efficiency and security requirements were developed, usually based on the EGL version of OT concept. This version lies behind many recent implementations of Yao's Garbled Circuits protocol for secure two-party computations, where oblivious transfers are used for getting the garbled values of users' input. Also, on $1 - out - of - 2$ OT are based also some generalizations, such as $1 - out - of - n$ OT and $k - out - of - n$ OT, introduced for applications to more advanced problems such as secure multiparty computations.

None of the currently known OT protocols is developed from scratch – all of them are constructed on some assumptions.

2.1.1 Rabin's OT

The oblivious transfer protocol by Rabin was introduced as a tool for two parties committing an exchange of secrets. In the OT one party appears as sender and the other one as receiver. The sender sends the product of two primes chosen by himself to receiver and after the interaction the receiver gets

the factorization of the number received with probability $\frac{1}{2}$. This problem assumes that factorization is a hard problem to solve, so the receiver cannot compute the factorization having the product. The protocol is presented below:

OT Protocol.

Step 1:

Sender chooses large primes p and q, computes and sends $n_A = p * q$ to Receiver.

Step 2:

Receiver chooses $r_B < n_A$ randomly, computes $c = r_B^2 \ mod \ n_S$ and transfers to Sender.

Step 3:

Sender finds an r_S such that $r_S^2 = c \ mod \ n_S$ (for calculation of r_S sender uses a square root extraction algorithm using the fact that she has the factorization of n_S [11]) and sends r_S to Receiver.

Receiver computes $g = gcd(n_S, r_R - r_S)$ (greatest common divisor).

$$\Pr[g = p \ or \ g = q] = \frac{1}{2},$$

So Receiver will find out the factorization of n_S with $\frac{1}{2}$ probability and Sender will stay unaware whether the factorization was found out.

2.1.1.1 EOS Protocol

Rabin constructed a protocol for exchange of secrets (EOS) based on the one presented below also presented in [10]. It allows two parties to pass their secrets to each other with at success possibility of $\frac{3}{4}$. Here we will present the

EOS protocol of Rabin also. Consider two parties A and B holding secret bits S_A and S_B respectively, want to exchange their secret bits. The secrets are passwords to files F_A and F_B, which are destroyed, when wrong password is used for access. Also, A is notified if F_A is accessed, and B is notified if F_B is accessed. Files aren't accessed by chance, so if F_A is accessed, then B definitely found out A's secret and also, passwords are used instantly, so if B finds out S_A, A becomes aware of it. In fact we can allow file access by chance, if we assume that the file owner is notified whether the file was destroyed or not.

Each of the parties also holds a public key for encrypting and signing messages. A signs each message sent to B with K_A and B signs with K_B. As a first step A and B invoke OTs using the protocol described above. We denote with p_A, q_A the primes chosen by A and with p_B, q_B the ones chosen by B.

$$n_A = p_A * q_A \text{ and } n_B = p_B * q_B.$$

After the oblivious transfers were made the probabilities for B knowing the factorization of n_A and A knowing the factorization of n_B are both $\frac{1}{2}$. We indicate with ϑ_B if B found out the factorization of n_A:

$$\vartheta_B = \begin{cases} 0 & if\ \gcd(n_A, r_A - r_B) = p_A\ or\ q_A \\ 1 & otherwise \end{cases},$$

where r_B is the randomly chosen number by B(as receiver) during the Step 2 and r_A is the number computed by A(as sender) during Step 3 of OT protocol. ϑ_A is defined similarly for the opposite OT invocation with B as sender and A. B computes the XOR of its secret with the indicator of n_A's factorization

knowledge $\varepsilon_B = S_B \oplus \vartheta_B$ and sends it in a signed message to A. A computes and sends ε_A to B in a similar manner.

Finally, A generates a random message m_A and places S_A as in the center of that message. After, A sends $E_{n_A}(m_A)$ in a signed message to B, where E_{n_A} is the encryption algorithm (with n_A as key) of any public-key encryption scheme, where the factors of n_A are required for decryption. Similarly, B sends $E_{n_B}(m_B)$ to A.

Theorem:

The probability of failed secrets' exchange with the EOS protocol is $\frac{1}{4}$.

Proof. If A gets notification that F_A is accessed, she deduces that B knows the factorization of n_A(he wouldn't attempt otherwise), so $\vartheta_B = 0$, which means that A also finds out S_B, because she has

$$\varepsilon_B = \vartheta_B \oplus S_B = S_B.$$

This means, that the secrets are exchanged even if only one of the participants found out the factorization of the number it received.

As, the probability of neither side knowing the factorization due to OT protocol is $\frac{1}{2}$, so the probability of the EOS protocol fail is $\left(\frac{1}{2}\right)^2 = \frac{1}{4}$. ■

2.1.2 EGL 1-out-of-2 oblivious transfer

In [12] Even et al. developed another version of oblivious transfer which had a different way of operation compared to Rabin's definition. Here the protocol is, again, executed between two parties, one of which acts as sender and the other one as receiver. In Rabin's version the sender sent information to

receiver and stayed oblivious whether the latter got it right (found the factorization) or not and the receiver didn't have any impact on the final action. The difference here is that the sender definitely knows that the receiver got some information, but she is not aware what information exactly she got. Also in this variant the sending party has two messages, one of which the receiver eventually gets, using his own input to the protocol. Most importantly, the information transmitted via Rabin's protocol is factorization, while here the messages can be chosen arbitrarily. The point is that in the EGL version also known as $1-out-of-2$ oblivious transfer, the sender inputs two messages (m_1, m_2) into the protocol. The outputs of the protocol is that the sender outputs m_s, where s is available exclusively to the receiving party only.The latter outputs nothing (see figure). The oblivious nature of the protocol is that the receiver gets no information about m_{1-s} and the sender stays oblivious about which message she has sent i.e. the selection index s.

Interestingly, the Rabin OT can be implemented easily using any $1-out-of-2$ OT protocol. The point is that in the latter case the sender stays oblivious about the selection index i.e. after the protocol execution the sender can make a guess on it with probability of $\frac{1}{2}$. To construct Rabin's OT from this one we just need to make one of the messages in possession of sender public. After the protocol execution the receiver, who doesn't know which of the messages she knows, will get the secret message with $\frac{1}{2}$ probability.

Now we describe the EGL OT protocol defined generically, after which we will present the Naor-Pinkas $1-out-of-2$ OT protocol[19] as a specific implementation. The generic nature of this protocol is that even it requires any public-key scheme encryption scheme having two binary operations $x \boxplus y$ and $x \boxminus y$ defined on its message space M_m as follows:

- The association $x \rightarrow x \boxplus y$ is a permutation on M_m, for every x.
- The association $y \rightarrow x \boxplus y$ is a permutation on M_m, for every y.
- $(x \boxplus y) \boxminus y = x$, for every x and y.

The operations can be defined in any way, while satisfying the above conditions. In case of using RSA[17] as the underlying encryption scheme, the above operations can be defined as:

$$x \boxplus y = (x + y) mod\ N, x \boxminus y = (x - y) mod\ N,$$

Where N is the modulus of RSA. In case of schemes with message space consisting from all Boolean vectors, bitwise XOR might be used for both operations.

EGL OT_2^1 Protocol

Inputs:

Sender: secret messages M_0, M_1

Receiver: selection bit σ

Step 1:

Sender chooses an encryption scheme (E_m, D_m) with message space M_m, where both E_m and D_m are permutations on M_m , along with two random messages $m_0, m_1 \in M_m$ and sends the three items to Receiver.

Step 2:

Receiver randomly chooses $r \in \{0,1\}$ and a message $k \in M_m$ and sends q to Sender, where

$$q = E_m(k) \boxplus m_r$$

Step 3:

Sender computes $k_i = D_m(q \boxminus m_i)$, for $i \in \{0,1\}$ and sends m'_0 and m'_1, and randomly chosen value s to Receiver, where

$$m'_0 = M_0 \boxplus k_s, \qquad m'_1 = M_1 \boxplus k_{s\oplus 1}$$

The above protocol works correctly in the semi-honest model i.e. when both parties follow the protocol. [12] also shows cheating from both parties can be detected by other participants with a probability of $\frac{1}{2}$. After the execution Receiver is able to read one of the secret messages (more specifically $M_{s\oplus r}$) and has no information on the other message and Sender is unaware of which message she has read.

$$m'_{s\oplus r} = M_{s\oplus r} \boxplus k_{s\oplus(s\oplus r)} = M_{s\oplus r} \boxplus k_r$$

From *Step 3* we have

$$k_r = D_m(q \boxminus m_r) = D_m(E_m(k) \boxplus m_r \boxminus m_r) = D_m(E_m(k)) = k.$$

k was chosen by the receiver in *Step 2*, so she can compute

$$M_{s\oplus r} = m'_{s\oplus r} \boxminus k.$$

So receiver should compute $m'_i \boxminus k$ for $i \in \{0,1\}$ and recognize the secret message from those values she got. The last statement is valid if the messages used satisfy the definition of recognizable secret with respect to a one-way function f, which is:

Definition: A message m from message space M_m is a recognizable secret to f if $f(m)$ is the only information known about M.

So having the values of the one-way function on secret messages, receiver can recognize a message from a random sequence.

Interestingly, s sent in Step 3 to Receiver can be used by the latter for finding out the index of message received, so its transmission can be avoided, if that information is not required.

2.1.3 NP 1-out-of-2 oblivious transfer

Another improvement to EGL OT protocol is the Naor-Pinkas 1-out-of-2 OT (a revision of [22]), which is now used as basic protocol in many applications and enhancing protocols[20, 21]. The distinctiveness of this protocol lies behind the capability of Receiver for deciding upon the index of Sender's input message she prefers to get. Actually, this feature can be also implemented with slightly changing the EGL 1-out-of-2 OT protocol so that the Sender generates and sends s to Receiver on an earlier stage, so that the latter choses r in a way to make $s \oplus r$ his desired index. This property can be very useful for straightforward applications, where the receiver doesn't know the information items Sender possesses, but has some meta-information, that helps to decide which item she needs. To conclude, here Sender has two information items/secret messages M_1 and M_2, only one of which he is ready to share with Receiver, but on the other hand, the receiving party has the capability to choose with his selection bit which of the messages she wants to get, but doesn't want Sender to become aware of it, while providing the required item. Also, this protocol by Lindell and Pinkas has a two-stage operation and doesn't require existence of a random oracle. The security of this protocol relies on decisional Diffie-Hellman computational hardness assumption, which states, that for a group G which is cyclic and has order n and generator g, having the g^a and g^b, where $a, b \in Z_n$ are chosen randomly, g^{ab} is computationally

indistinguishable from g^c, where $c \in Z_n$ is also chosen randomly. Below we present the protocol.

NP OT_2^1 Protocol.

Inputs:

Sender: secret messages m_0, m_1

Receiver: selection bit σ

Auxiliary input:

Both Sender and Receiver share knowledge of a group G of order n and a generator g for that group. We will also assume, that m_0 and m_1 are associated with some elements of G.

Step 1:

Receiver randomly chooses three integers $a, b, c \in [0, n-1]$ and composes a tuple λ depending on the value of σ:

In case $\sigma = 0$, $\lambda = (g^a, g^b, g^{ab}, g^c)$.

In case $\sigma = 1$, $\lambda = (g^a, g^b, g^c, g^{ab})$.

After constructing λ, she sends it to Sender.

Step 2:

As g^a, g^b, g^c and g^{ab} are some integers, we denote the received tuple as (x, y, z_0, z_1). Note, that

$$z_\sigma = g^{ab}, \qquad z_{\sigma \oplus 1} = g^c.$$

Sender randomly chooses $u_0, u_1, v_0, v_1 \in [0, n-1]$ and computes the below values for l_0, l_1, k_0, k_1:

$$l_0 = x^{u_0} * g^{v_0}, \qquad l_0 = x^{u_1} * g^{v_1}$$

$$k_0 = z_0^{u_0} * y^{v_0}, \qquad k_1 = z_1^{u_1} * y^{v_1}$$

The values k_0 and k_1 are used to encrypt the secret messages, so Sender computes using G's multiplication operator $c_0 = m_0 * k_0$ and $c_1 = m_1 * k_1$, as m_i, k_i-s are G's elements, for $i \in \{0,1\}$ and transmits pairs (l_0, c_0) and (l_1, c_1) to Receiver as the finalization of the Step 2.

The information Receiver got through communication, along with her own parameters are sufficient for computing the desired value of m_σ with formula

$$m_\sigma = c_\sigma * (l_\sigma^b)^{-1}.$$

As $c_\sigma = m_\sigma * k_\sigma$, we only need to show, that $l_\sigma^b = k_\sigma$, where $k_\sigma = z_\sigma^{u_\sigma} * y^{v_\sigma}$.

$$l_\sigma^b = x^{u_\sigma * b} * g^{v_\sigma * b} = g^{u_\sigma * a * b} * g^{v_\sigma * b} = z_\sigma^{u_\sigma} * y^{v_\sigma} = k_\sigma.$$

So $c_\sigma * (l_\sigma^b)^{-1} = c_\sigma * k_\sigma^{-1} = m_\sigma * k_\sigma * k_\sigma^{-1} = m_\sigma$ ■

2.1.4 NP 1-out-of-n oblivious transfer

The OT protocols described in previous paragraph handle scenarios with two parties, one of which holds two secrets which are two large primes in case of Rabin's protocol, recognizable secret messages in case of Even's et al. 1-out-of-2 protocol and secret messages in case of Naor-Pinkas 1-out-of-2 protocol. But in real life applications there are situations, where the sender holds more than two secrets and the receiver stills needs one of the secret messages. Here we will describe a conceptual development of the oblivious transfer protocol introduced by Brassard, Crépeau and Robert in [24] known as All-or-Nothing

Disclosure of Secrets (ANDOS) or $1-out-of-n$ oblivious transfer protocol. As might be expected, this flavor of OT handles the scenario with multiple secret messages input from Sender.

We again, consider two parties one of which (denoted as Sender) has n secret messages and wishes the other party (denoted as Receiver) to gain no information but one secret message. Receiver, on the other hand, wants to obtain a certain element from the possession of the Sender, but doesn't want to disclose any information about the element received to Sender. Sender is assumed to be honest in terms of evading the transmission of the secret messages, but she is assumed to be interested in the selection index of Receiver.

The protocol we present below is also implementing $1-out-of-n$ oblivious transfer functionality. It was introduced in [19] and is considered a starting point or for many variations invented later. This protocol doesn't require for the sender to have a publicly available value or public-key. Instead it requires a semantically secure encryption system similar to ElGamal [27] and its security relies on the decisional Diffie-Hellman computational hardness assumption [23]. The protocol also avoid random oracle model by using probabilistic hashing instead of random oracle, in order to map the secret messages to the subgroup of protocol operation. For the sake of the simplicity, we will assume that the secrets are from G.

NP $1-out-of-n$ OT

Inputs:

Sender: n secret messages $m_1, m_2, \ldots, m_n$.

Receiver: selection index $\sigma \in [1,n]$

Auxiliary input:

Both Sender and Receiver share knowledge of a group G of order n and a generator g for that group.

Step 1:

Receiver randomly generates g^a, g^b and chooses c_0, c_1, so that $c_\sigma = ab$ and $c_{1-\sigma}$ is chosen randomly. A tuple (x, y, z_0) is transmitted to Sender, where

$$x = g^a, \qquad y = g^b, \qquad z_0 = g^{c_\sigma - \sigma}$$

Step 2:

As g^a, g^b, g^c are some integers, we denote the received tuple as (x, y, z_0). Note, that

$$z_\sigma = g^{ab}, \qquad z_j = g^c * g^j, for\ 1 \leq j < n.$$

Sender randomly chooses $u_i, v_i \in [0, n-1], for\ 0 \leq i < n$ and computes the below values for l_i, k_i, for $0 \leq i < n$:

$$l_i = x^{u_i} * g^{v_i}, k_i = z_0^{u_i} * y^{v_i} \text{ for } 0 \leq i < n.$$

The values k_i for $0 \leq i < n$ are used to encrypt the secret messages, so Sender computes using G's multiplication operator $c_i = m_i * k_i$, as m_i, k_i-s are G's elements, for $0 \leq i < n$ and transmits pairs (l_i, c_i) to Receiver as the finalization of the Step 2.

After getting the pairs transmitted by Sender in Step 2, Receiver is able to correctly compute the desired secret message in a similar manner to the NP OT_2^1 with the formula $m_\sigma = c_\sigma * (l_\sigma^b)^{-1}$.

2.2 White-box oblivious transfer

A novel approach of the oblivious transfer problem was demonstrated by Jivanyan and Khachatryan in [20]. Two new protocols based on the concepts of

$1-out-of-2$ and $1-out-of-n$ OT were introduced with their implementations having dramatically enhanced performance and require less communication between parties. The main reason lying under the improvement is elimination the use of traditional public-key cryptosystems. Instead, here the encryptions are made with use of white-box cryptographic (WBC) techniques. Previously, WBC was mainly used as a tool for software protection and digital rights management [47].

With a white-box implementation of a symmetric key cryptographic scheme one can spread the encryption table of his scheme, so that everyone can encrypt messages, but while the secret key and/or the decryption table are in his possession, no one is able decrypt those messages but the owner if the scheme is secure.

Below two protocols are presented inspired by EGL OT_2^1 and NP OT_n^1. In protocols we denote with Gen the algorithm responsible for generating white-box tables for selected secret key with the underlying encryption scheme. Enc_T stands for the white-box encryption algorithm using tables T. Dec is the decryption algorithm of underlying encryption scheme, using a secret key.

2.2.1 1-out-of-n WBOT

Inputs:

Sender: n l-bit secret messages $M_1, M_2, \ldots, M_n$, where l and is known to the Receiver.

Receiver: selection index $\sigma \in [1,n]$

Pre-computation:

Sender generates a secret key k and with the use of tables generation algorithm Gen constructs $T = Gen(k)$ – the white-box encryption tables and sends T to Receiver.

Step 1:

Sender generates n random values m_1 , $m_2, \ldots, m_n \in \{0,1\}^l$ and sends them to the Receiver.

Step 2:

Receiver generates random $r \in \{0,1\}^l$ and encrypts it with help of the white-box encryption tables T. The encrypted value is XOR-ed with m_σ and the result v to Sender, where $v = m_\sigma \oplus Enc_T(r)$.

Step 3:

Sender computes n candidates $r_1, r_2, \ldots , r_n$ of r, where

$$r_i = Dec_s(v \oplus m_i), \qquad i \in [1, n],$$

and transmits n messages $M'_1, M'_2, \ldots, M'_n$ to Receiver, where

$$M'_i = M_i \oplus r_i, \qquad i \in [1, n].$$

Receiver computes $M_\sigma = M'_\sigma \oplus r$.

2.2.2 1-out-of-2 WBOT

Inputs:

Sender: 2 l-bit secret messages M_1, M_2 where l and is known to Receiver.

Receiver: selection bit $\sigma \in \{0,1\}$

Pre-computation:

Sender generates a secret key k and with the use of tables generation algorithm Gen constructs $T = Gen(k)$ – the white-box encryption tables and sends T to Receiver.

Step 1:

Sender generates a value $c \in \{0,1\}^l$ randomly and transmits it to Receiver.

Step 2:

Receiver generates a value $r \in \{0,1\}^l$ randomly and transmits m_0 to Sender, where m_0 is one of the values defined the following way:

$$m_\sigma = Enc_T(r), \qquad m_{1-\sigma} = c \oplus Enc_T(r).$$

$Enc_T(r)$ is the encryption of r with the white-box tables T.

Step 3:

With use of m_0 Server computes $m_1 = m_0 \oplus c$ and transmits two messages M_1', M_2' to Receiver, where $M_i' = M_i \oplus Dec_s(m_i),\ i \in \{0,1\}$.

Receiver computes $M_\sigma = M_i' \oplus r$.

2.3 WBOT extension protocols

Here we will show the construction of extension for white-box cryptography based oblivious transfer protocol. These protocols are considered in the random oracle model. $H: [m] \times \{0,1\}^k \to \{0,1\}^m$ is a correlation robust hash function and $G: \{0,1\}^k \to \{0,1\}^m$ is a pseudorandom number generator, where k is a security parameter defined preliminarily.

Definition: The m-times $1 - out - of - n\ WBOT$ functionality for l-bit vectors. denoted $m \times WBOT_l$, is defined in the following way: The sender S holds m pairs of vectors $\{x_j^0, x_j^1\}$. The receiver R has m selection bits $(r_0, r_1, \ldots, r_{m-1})$. After the invocation of the protocol R should have m

vectors $\{x_j^{r_j}\}$ while staying oblivious of the other vectors and R stays unaware of the selection bits.

The Protocol 0 presented below replaces $m \times WBOT_l$ with $k \times WBOT_l$, resulting the same output with less l-bit $WBOT$ invocations.

Protocol 0:

Inputs:

S: m pairs of l-bit vectors (x_j^0, x_j^1)

R: vector of selection bits $r = (r_0, r_1, \ldots, r_{m-1})$

Step 1:

S initializes a k-bit vector $s$randomly

R initializes a $m \times k$matrix $T$with the random oracle

Step 2:

$WBOT$ is invoked where R has the role of sender inputting $(t_i, t_i \oplus r)$, where $0 \leq i < k$ and S has the role of receiver with input s.

Step 3:

S sends $m$pairs of l-bit vectors (y_j^0, y_j^1) to R, where

$y_j^0 = x_j^0 \oplus H(j, q_j)$, $and\ y_j^1 = x_j^1 \oplus H(j, q_j + s)$

Where q_j is the j^{th}row of the matrix received by S in Step 2 with columns q^i

R outputs $z_j = y_j^{r_j} \oplus H(j, t_j)$.

Fig. 10: WBOT extension intermediate tables

Protocol 1 replaces $m \times WBOT_l$with $k \times WBOT_k$, resulting in reduction of both, $WBOT$ invocation count and length.

Protocol 1:

Inputs:

S: m pairs of l-bit vectors (x_j^0, x_j^1)

R: $r = (r_0, r_1, \ldots, r_{m-1})$

Step 1:

S initializes a random k-bit vector s

R initializes k pairs of random k-bit vectors $\{k_i^0, k_i^1\}$

$WBOT$ is invoked where R plays sender with inputs $\{k_i^0, k_i^1\}, i \in [0, k)$ and S plays receiver with input s.

Step 2:

$m \times k$ matrix T is constructed, where $t^i = G(k_i^0)$

R sends $u^i = t^i \oplus G(k_i^1) \oplus r$ to S for all $i \in [0, k)$

S defines $m \times k$ matrix Q, where $q^i = s_i * u^i \oplus G(k_i^{s_i})$

Step 3:

S sends m pairs of vectors (y_j^0, y_j^1) to R, where

$y_j^0 = x_j^0 \oplus H(j, q_j)$ and $y_j^1 = x_j^1 \oplus H(j, q_j \oplus s)$

R outputs $z_j = y_j^{r_j} \oplus H(j, t_j)$

Provided with WBOT proved secure with semi-honest participants (both parties strictly follow the protocol, but try get more information "legally")[14] it is easy to show, that these extensions are secure in random oracle model. Below is provided the proof of correctness of Protocol 1. Protocol 0 can be proved correct with similar considerations.

2.3.1 Proof of correctness

What we need to prove is that $z_j = x_j^{r_j}$

first of all, let's find a relation between $q_j$and t_j

in Step 2 $q^i = s_i * u^i \oplus G(k_i^{s_i})$, $u^i = t^i \oplus G(k_i^1) \oplus r$ and $t^i = G(k_i^0)$.

from these three formulas we get:

$$q^i = s_i * (G(k_i^0) \oplus G(k_i^1) \oplus r) \oplus G(k_i^{s_i})$$
$$= s_i * G(k_i^0) \oplus s_i * G(k_i^1) \oplus s_i * r \oplus G(k_i^{s_i})$$

when $s_i = 0$,$s_i * G(k_i^0) \oplus s_i * G(k_i^1) \oplus G(k_i^{s_i}) = G(k_i^0)$,

when $s_i = 1$,$s_i * G(k_i^0) \oplus s_i * G(k_i^1) \oplus G(k_i^{s_i}) = G(k_i^0)$, so

$$q^i = s_i * r \oplus G(k_i^0) = s_i * r \oplus t^i$$

therefore $q_j = s * r_j \oplus t_j$.

now let's apply this result to $z_j = y_j^{r_j} \oplus H(j, t_j)$

case 1: $r_j = 0$

$z_j = y_j^0 \oplus H(j, t_j)$and from Step 3, $y_j^0 = x_j^0 \oplus H(j, q_j)$, so

$$z_j = x_j^0 \oplus H(j, q_j) \oplus H(j, t_j) = x_j^0 \oplus H(j, s * r_j \oplus t_j) \oplus H(j, t_j)$$
$$= x_j^0 = x_j^{r_j}$$

case 2: $r_j = 1$

$z_j = y_j^1 \oplus H(j, t_j)$and from Step 3, $y_j^1 = x_j^1 \oplus H(j, q_j \oplus s)$, so

$$z_j = x_j^1 \oplus H(j, q_j \oplus s) \oplus H(j, t_j)$$
$$= x_j^1 \oplus H(j, s * r_j \oplus t_j \oplus s) \oplus H(j, t_j) = x_j^1 = x_j^{r_j}$$

from both cases we get $z_j = x_j^{r_j}$, so $Protocol\ 1$works correctly. ■

2.3.2 Proof of security

We will show the secureness of the sender and the receiver separately lying on the security of the underlying oblivious transfer protocol, which is, in our case secure in the semi-honest adversarial model.

Receiver's security: All the information available to the sender appearing to be corrupted during the protocol execution are the intermediate u^i-s and also the k oblivious transfer execution outputs, where he plays the receiver.

In simulated mode we assume s chosen randomly by Sender is uniform, from $\{0,1\}^k$ are chosen k bit vectors $\{k_i^{s_i}\}$ uniformly and, finally, from $\{0,1\}^k$ m-bit vectors $\{u^i\}$ are also chosen uniformly.

In the real execution s and $\{k_i^{s_i}\}$ are chosen uniformly (as done in the simulated mode). The intermediate messages u^i are selected as $u^i = G(k_i^0) \oplus G(k_i^1) \oplus r$, and with respect to the security of underlying oblivious transfer protocol, the Sender doesn't know $k_i^{1-s_i}$, therefore, $G(k_i^{1-s_i})$ is cannot be revealed, so the information available to the Sender in real and simulated executions cannot be distinguished.

Sender's security:

All the information available to the sender appearing to be corrupted during the protocol execution are the message pairs $\{(y_i^0, y_i^1)\}$ sent to him in the last step of the protocol.

In simulated mode we assume the corrupted receiver to choose the chooses the matrix T along with the values $\{(k_i^0, k_i^1)\}$, computes the values $y_j^{r_j} = x_j^{r_j} \oplus H(j, t_j)$ and chooses the other values $y_j^{1-r_j}$ uniformly. As the $y_j^{r_j}$-s are computed similarly in real and simulated modes, the receiver gains no advantage here. The weak point could be the values $y_j^{1-r_j}$, but with respect to the used correlation robust hash function, those values are either indistinguishable in real and simulated execution modes.

Chapter 3. SMC Framework using WBOT

Overview

This chapter contains the description, analysis and experimental results for our secure multiparty computation framework presented in [50, 51, 53]. Here are presented some existing implementations of SMC frameworks in comparison to our approach. Building blocks for our framework are described along with the framework lifecycle description and secure multiparty computation protocol by Goldreich, Micali and Widgerson (GMW). This chapter also contains description of compiler into our framework for generating single pass Boolean circuits from higher level function description. The chapter is finalized with presentation of our modified version of GMW protocol enabling participation of parties passive in terms of their engagement in the computation process. The protocol is justified with an application example for combining competitor service providers into a single ordering system in a privacy preserving manner.

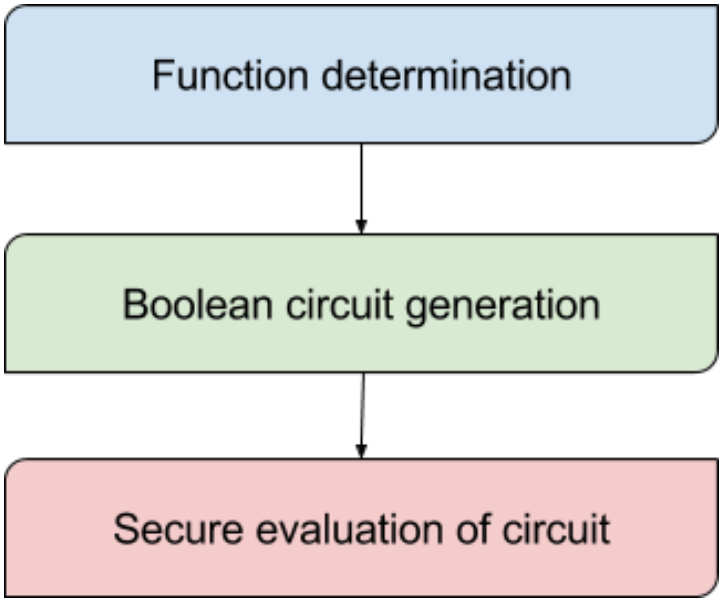

Fig. 11: Lifecycle of the SMC framework.

Most existing implementations of secure multiparty computations rely on an arithmetic-circuit representation of the function, where VIFF [89] is a prominent example with comparably good performance and downloadable code. There are also Boolean circuit based approaches to the problem, most known of which are FairplayMP [54] and GMW protocol implementation by Choi et al., which will is taken as a comparison, because the available implementation of FairplayMP has input size limited to 16 bits and each party limited to only one input. VIFF and other arithmetic circuit based frameworks are suitable for outsourcing applications, where the computing parties have different computing power and also are reportedly slower, than Choi et al. GMW framework.

3.1 Compiler

GMW as well as several other secure computation protocols, takes as an input the Boolean representation of the function, which is intended to be computed [1, 30]. Some of those protocols were implemented during recent years, descriptions of which can be found in [31, 32] and elsewhere. Along with the framework implementations, several application scenarios based on those frameworks were implemented and even embedded in real life applications, some of which are computing market clearing prices in Denmark [35], auctions [36], contract signing [37], etc. All of the implemented frameworks have dependence in terms of execution time and communication cost on the circuit dimensions of the representation of the desired function, whatever application it is intended to be used for. However, building a binary or an arithmetic circuit implementing the desired functionality is not a trivial task, as it requires expert knowledge in circuit construction, verification and the exclusive properties of the protocol being used even if we ignore the huge size of circuits implementing non-trivial functions. There are software tools for electronic design automation, that can handle these tasks, but the applications

of our framework are not limited to being designed by computer scientists or EDA experts, so our framework includes a compiler, which is responsible for the generation of Boolean circuit which will implement the function provided in a special high level programming language as an input to it.

The first ever implemented secure computation framework, intended for secure two-party computations Fairplay [37] includes a compiler that generates Boolean circuits based on the high level description of the desired function. For the function description Secure Function Description Language (SFDL) [38] is used. As this was the first implementation of such framework with the compiler also being a unique development, it has some problems with efficiency, because it takes a lot of time to generate a Boolean circuit even for not complicated functions. In some cases the compiler is also unreliable, because during the testing of it we faced termination of processing of AES-128, which is a very basic operation to compute securely. On a machine with 48GB random access memory out-of-memory error is reported after 23 minutes of processing in [55]. So, for being independent from implementation problems and inefficiency of the existing Boolean circuit generator, we implemented our compiler, which worked perfectly on the circuits the Fairplay compiler failed even with computers with much lower computational resources [51]. The Boolean circuit generation for AES-128 was handled with 8GBs of random access memory (six times less) on a i5-5200U 2.2 GHz CPU [51, 55].

Our purpose during the development of the compiler was to make it applicable for any secure computation framework (specifically, we have different modes of operation for circuit generation for garbled circuit protocol and for GMW protocol). We allow the users to configure the compiler greatly for resulting in more efficient circuit constructions for specific applications, but the compiler itself is general purpose and does not concentrate on specific protocol or application, like Huang et al. implemented in [39]. Although, the experiments carried out on the compiler were done through secure multiparty computation

protocol of Golreich et al., which is the underlying protocol of the framework implementation presented in this thesis and also with Yao's Garbled circuits protocol, which is the underlying protocol of the framework implementation presented in cooperation with Sokhakyan presented in paper [50, 51, 55].

The two approaches differ greatly in terms of priorities of the generated circuits' size and depth properties. Specifically, the GMW protocol works faster on low depth circuits, even if the overall count of the gates is slightly larger, but in case of Yao's protocol, the circuit depth has no significance, so we configure the compiler properly, for generating convenient circuits for both cases.

Although many techniques were obtained from hardware design principles for circuit constructions, the circuits used for secure computations differ dramatically from those. The main point of the generated circuits is that they have to be single pass and not change values of any wire during the execution. The single-pass property of the Boolean circuits is taken into account in the designs of secure computation protocols. We used special circuit constructions for generation low-depth Boolean circuits described in detail in section 3.2.3.3. To be more specific, low-depth sub-circuits are preferred over low-size sub-circuits, some of which are demonstrated in Table 1.

The GMW, as well as garbled circuits protocol spend very little computational resources on computation of XOR gates (usually they are considered to be evaluated "for free"). The case of AND gate is radically different, because most of the execution time spent during evaluation with GMW protocol is spent on the computing of these gates, so the less AND gates are present in the circuit, the less is the execution time. A figure demonstrating two circuits with different number of AND gates implementing same operation is demonstrated in Fig. 12.

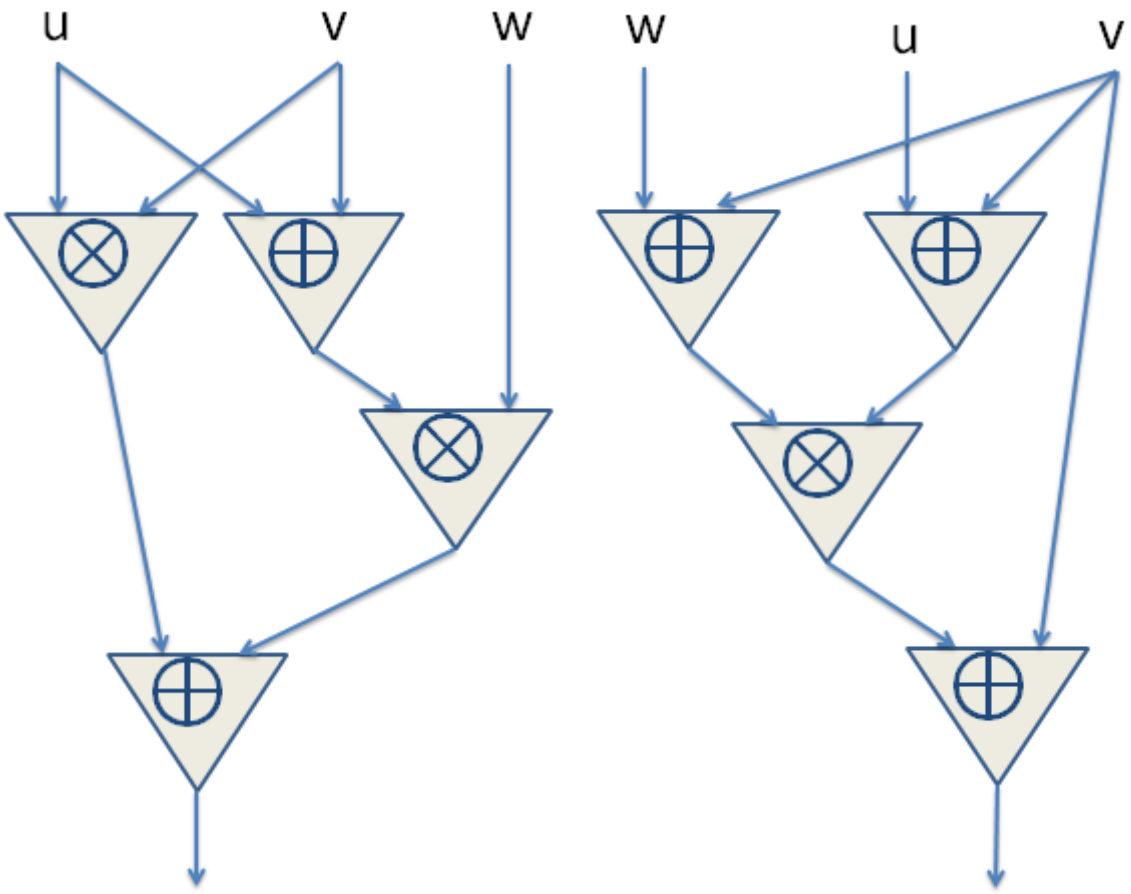

Fig. 12: Two alternative circuits computing function (c⊕b)⊗(a⊕b)⊕b

3.1.1 Input language

Here we will provide a brief description of the input language of our compiler. The inspiration and base of the language we use is the Secure Function Definition Language (SFDL) [38]. Our input language is capable to avoid function composition for computing primitive functions – here those can be described in global space. With several modifications this language handles definition of very complicated functionalities.

The desired functionality can be described in several files, which are being included into each other. However, the final processing is carried out upon one file, containing the whole descriptions. The included files may have variables with coinciding names, which can result in error prone situations. For this reason, we rename all variables, when including new files to not have name collisions.

We also have several modifications against SFDL regarding supported operations. The most prominent difference from SFDL is that the high-level programming language described here has no variable types, so they are being declared by simply mentioning them, in contrast to SFDL, where there's certain syntax defined, which is compulsory to use for variable declaration.

The condition operations are supported in the traditional manner:

- $if \ < expr\_1 > \ then$

  $< expr\_2 >$

  end

- $if \ < expr\_1 > \ then$

  $< expr\_2 >$

  $else$

  $< expr\_3 >$

  end

where $< expr\_1 >$ is a valid Boolean expression and $< expr\_2 >$ with $< expr\_3 >$ are sets of valid language expressions.

As the generated circuit should be a single pass Boolean circuit, the variable assignments should be carried out in a similar way, which means that no variable changes its value, instead a new variable is being introduced each time an assignment is carried out.

To demonstrate this we will consider a trivial example of variable incrementing $a := a + 1$. In this case a new variable a' is introduced which carries the value $a + 1$.

Our language also supports iterations and function calls. As a single pass Boolean circuit should be constructed, we limit the number of iterations, so one could never produce an infinite loop.

The difference from SFDL regarding function declarations is that we are able to present only return type for the function. The function calls are placed inline, whenever available just like as C++ template function instantiation, which is thoroughly discussed in [81].

Input and output in the high level language of our compiler are emphasized with the method given below:

$defvar\ <var> := input.<p>\{<length>\}$,

where $<var>$ is the variable name, $<p>$ is the participant owning the input and $<length>$ is the input length.

$output.<p> := <expr>$,

where $<p>$ is the participant owning the output and $<expr>$ is a valid expression regarding considered language.

3.1.2 Implementation details

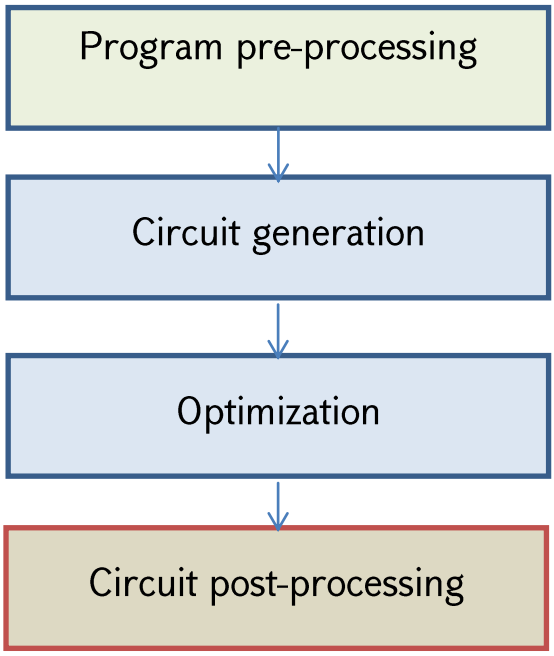

Fig. 13: Compilation stages

In the generation phase an temporary circuit implementing the function is constructed, which is usually very inefficient. Next comes the optimization phase, where the inefficient circuit is transformed to an efficient one by applying several optimizing techniques. And the last phase – post-processing,

is applied for GMW protocol specific circuit constructions and reorders gates in output files to enable better parallelization.

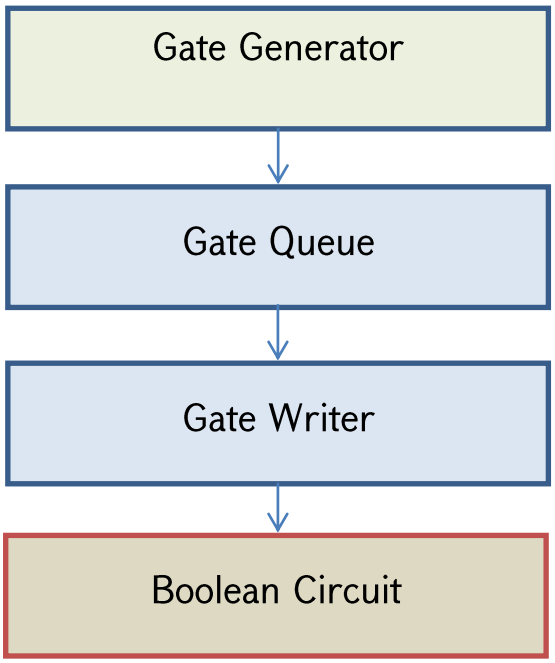

Fig. 14: Compiler circuit generator structure

The phase of circuit generation is divided into several sub-phases, as can be seen in Fig. 14. The gate generation stage is responsible for the initial gate construction, which are being passed to gate writer through a queue.
A major improvement upon the compiler included in Fairplay framework, our compiler doesn't keep the full circuit in the memory, which results in processing of large circuits with relatively modest RAM. Every phase of compiler operates on the circuit with a chunk-based approach. However, the more of the circuit is contained in memory, the more efficient the optimizations will be applied, so we enable configuring the compiler with providing maximum memory amount that is allowed to utilize. The best results in terms of time-complexity with different storage types was achieved while using memory mapped files.

3.1.3 Optimizations

Removal of inverters and identity gates

While generating the Boolean circuit for the desired functionality, the compiler may end-up generating inverters and identity gates. This can be present in situations, where in the high-level description of the function a variable is being incremented by constant or is a part of a logic expression. The identity gates are removed trivially, while the inverters are creating more problems. When removing those, we expect that some other gates may be transformed into inverters.

Removal of unused gates

As the last stage of front-end produced circuit optimizations, the gates having no impact on output wires generated during the generation phase are being removed from the circuit. Although keeping them will not result in any kind of malfunctioning of the circuit, they are totally needless, because of the lack of effect on the output wires. By removing these gates, we save precious computational resources that would be used for evaluation of those.

The circuit is being passed in down-up mode, meaning we take the output gates and pass to upper gates (marking them as alive) only when they are connected (have input) to gates being considered. When no continuation of the processing is possible, we stop and take to the final version only the gates being marked during the processing.

3.1.4 Post-processing stage for multiparty computations

An additional stage of circuit processing was added to make specific arrangements reasonable for secure multiparty computations only. The post processing unit modifies the output file structure in a way to enable efficient retrieval of gates on the same level, so that they can be processed in parallel and make possible efficient multithreaded execution of oblivious transfers for AND gate computations.

```
queue Q;
foreach (g in input_gates)
begin
    push(g, Q);
end
push(dummy_gate, Q)
while (empty(Q) == false)
do
    g = deque(Q);
    if (dummy(g))
    begin
        if (empty(Q) != false)
        begin
            push(g,Q);
        end
    end
    else
    begin
        foreach(s_g in successors(g))
        begin
            if (exported(s_g) == false)
            begin
                push(s_g, Q);
            end
        end
    end
    export(g);
end
```

Fig. 15: post-processing of Boolean circuit

The specifics of GMW protocol is that all AND gates require communication between parties for share based computations, and through the topologic order of the circuit, circuit evaluation cannot pass forward to gates if both inputs shares of those gates are not available (one or more of gates, upon whose output the inputs of the gate in question depends). As XOR gates are evaluated for free (no communication required), multithreaded evaluation of AND gates will speed up the circuit evaluation.

The purpose of this post-processing step is to sort gates in output file in a way to maximize the count of AND gates available at a time for evaluation, so we output gates in "levelled" order – no child gate appears before parent gate and dummy gates, are artificially added to the output file as markers for initiation of bulk evaluation, as in this sorting order inputs shares of gates listed between two markers are available for computation. Pseudo-code demonstrating high level implementation of the post-processing step is shown in Fig. 15.

The following definitions are used in code:

$successors(g)$ - set of successors of gate g (gates, where output wire of g is an input wire),

$input\_gates$ - set of input gates of circuit,

$dummy\_gate$ - marker for distinguishing levels of gates.

As seen in the pseudo-code from the figure, gates are being written to file in special layered order.

3.2 Multiparty computation module

3.2.1 Overview

As we have already mentioned in previous sections, secure multiparty computation (SMC) are intended to help several mutually distrustful parties wish to compute a common function without trusted parties, where inputs are provided by each of them and they do not want to disclose those inputs to other participants. A few protocols were suggested for solving the SMC problem. The approaches usually differ with the choice of the initial assumptions, underlying cryptographic primitives and security levels.

3.2.2 GMW protocol

The GMW protocol introduced by Goldreich, Micali and Widgerson in [1], is intended for computation of any function that is possible to represent as a single pass Boolean circuit, where the permitted operations (circuit gates) are XOR and AND. As the protocol avoids the engagement of trusted third parties, communication is required between all pairs of participants.

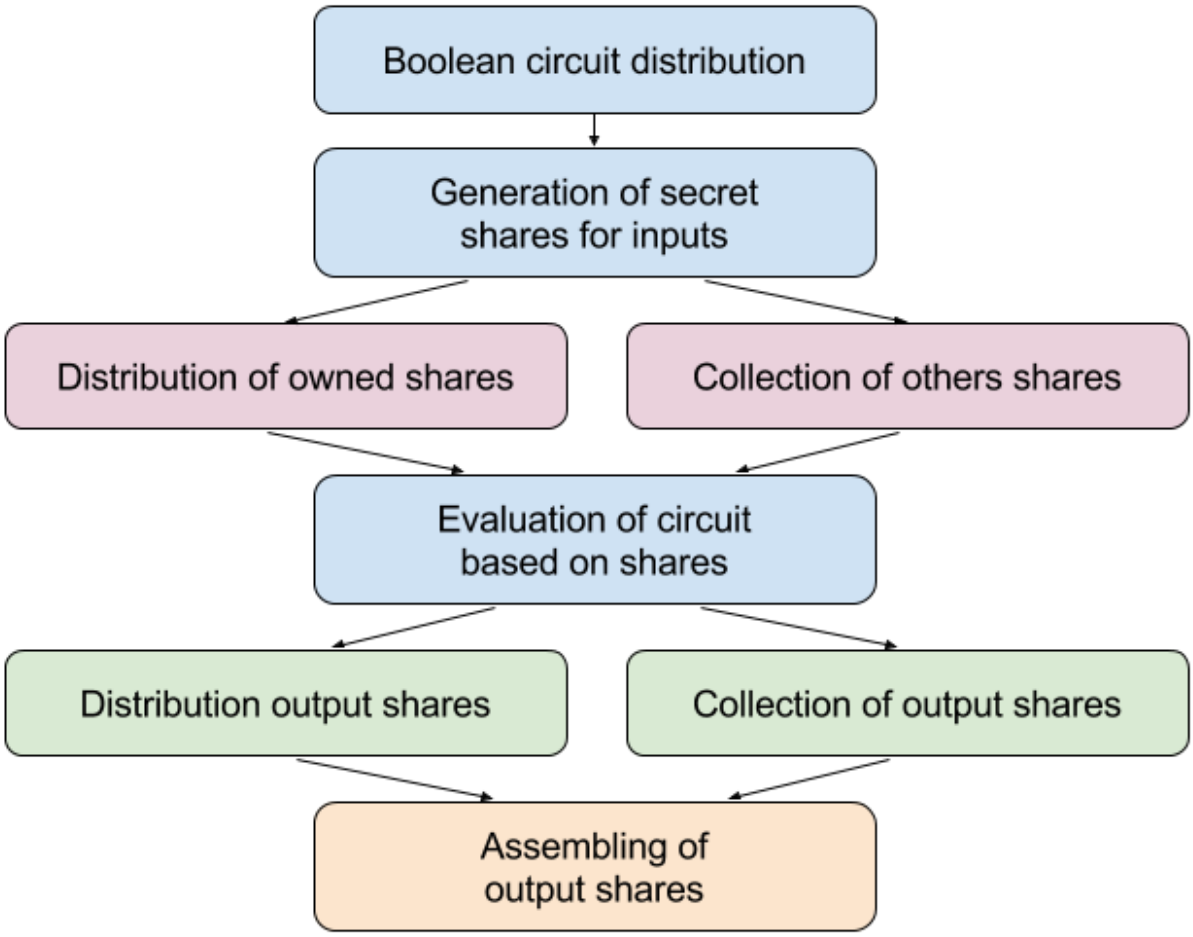

Fig. 16: Circuit evaluation flow from a participants' perspective

Unlike the Yao's garbled circuits protocol, where one party is the circuit generator and the other party is the circuit evaluator, here all the parties are possess the circuit locally and each of them plays an evaluator with input shares of himself and the other participants. Below we will present the building blocks of the protocol as secret sharing and 1-out-of-4 oblivious transfer, after which we will thoroughly analyze GMW protocol and each of its steps. At last

we will discuss optimization techniques we have applied in our implementation of this protocol and compare our performance to the others in terms of experimental results.

3.2.2.1 1-out-of-4 Oblivious Transfer

For implementing oblivious transfer protocol executions for the framework, we use white-box oblivious $1 - out - of - n$ described in section 2.2.1 in the extended version (section 2.3), to achieve best performance.

3.2.2.2 Secret Sharing

Each party generates n shares for each input wire w $(s_{w1}, \dots, s_{wn})$ randomly so, that $\bigoplus_{i=1}^{n} s_{wi} = s_w$. Share generation doesn't really require much effort to satisfy the above equation. Party P_j generates n-1 shares randomly for all other parties $P_i, i \neq j$. To comply to the equation, she just has to adjust its own share correspondingly –

$$s_{wj} = (\bigoplus_{\mathrm{i}\neq\mathrm{j}} s_{wi}) \oplus s_w.$$

After the share generation, each share s_{wi} is sent to corresponding participant P_i. Now, having shares for all input wires, each participant proceeds to circuit evaluation. After the evaluation, each participant P_i would have its own value s_{wi}, so for each output wire w of the circuit we again will have an n-tuple $(s_{w1}, \dots, s_{wn})$. To be combine those values into a single value for each wire the protocol suggests specific way of gate evaluation, according which the n-tuples of for each output wire will play as shares of parties, and the final value will be computed with formula

$$s_w = \bigoplus_{i=1}^{n} s_{wi}.$$

As the circuit consists of XOR and AND gates only, below we provide the methods for both gates' evaluation that should be used for all input, intermediate and output gates of the circuit and show they operate as expected.

3.2.2.3 Gate Evaluation

As already mentioned previously, the GMW protocol operates on a single pass Boolean circuit constructed with use of XOR and AND gates. Each participate holds her secret share for each input wire of the circuit and computes the circuit gate by gate using appropriate evaluation method. In the process of evaluation, one computes her share of values on all intermediate wires of the circuit. This means, that even in case of abortion of protocol in an intermediate phase, if one has the capability to collect the shares of wires already computed by all participants, she can get the true values on those wires by combining all shares. Below we present the evaluation methods of available gates with proof of their validities.

Evaluation of XOR gates

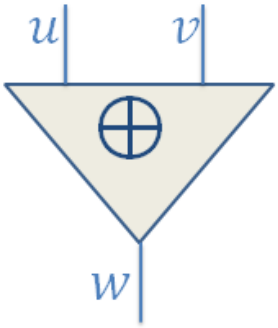

Fig. 17: XOR gate

Suppose we have an XOR gate with u and v input wires and an output wire w. All parties already have their shares for the values on input wires $(s_{u1}, \dots, s_{un})$ and $(s_{v1}, \dots, s_{vn})$. Each party P_i computes the value of the

output wire the following way: $s_{wi} = s_{ui} \oplus s_{vi}$. The n-tuple $(s_{w1}, \dots, s_{wn})$ would be a valid sharing for w's value, because:

$$s_w = \oplus_{i=1}^{n} s_{wi} = \oplus_{i=1}^{n} (s_{ui} \oplus s_{vi}) = (\oplus_{i=1}^{n} s_{ui}) \oplus (\oplus_{i=1}^{n} s_{vi}) = s_u \oplus s_v.$$

Evaluation of AND gates

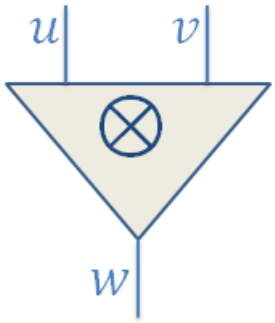

Fig. 18: AND gate

Suppose we have an AND gate with input wires u and v and an output wire w. The shares of gate's input values of u and v are $(s_{u1}, \dots, s_{un})$ and $(s_{v1}, \dots, s_{vn})$. The gate's output value s_w should be computed so, that

$$s_w = s_u \otimes s_v = (\oplus_{i=1}^{n} s_{ui}) \otimes (\oplus_{i=1}^{n} s_{vi}) =$$

$$= (\oplus_{i=1}^{n} (s_{ui} \otimes s_{vi})) \oplus (\oplus_{i<j} ((s_{ui} \otimes s_{vj}) \oplus (s_{uj} \otimes s_{vi}))).$$

The first sum $\oplus_{i=1}^{n} (s_{ui} \otimes s_{vi})$ can be computed locally for every participant, but for evaluation of the second sum one should communicate with other participants. To avoid collection of individual shares of other parties, oblivious transfer protocols are used for AND gate evaluation in the following way.

P_i has to interact with several P_j-s to compute each element of the sum

$$\oplus_{i<j} \left((s_{ui} \otimes s_{vj}) \oplus (s_{uj} \otimes s_{vi}) \right).$$

P_j generates a random value $r_j^{\{i,j\}}$ and composes four values with $r_j^{\{i,j\}}$ and its shares for input wires of the gate being computed s_{uj} and s_{vj}.

$$r_j^{\{i,j\}}, \quad r_j^{\{i,j\}} \oplus s_{uj}, \quad r_j^{\{i,j\}} \oplus s_{vj}, \quad r_j^{\{i,j\}} \oplus s_{uj} \oplus s_{vj}$$

After, the parties invoke a $1 - out - of - 4$ oblivious transfer with P_i as receiver and P_j as sender. P_i requests one of the abovementioned values with its index composed of its shares for the input wires s_{ui}, s_{vi}. P_i computes the required sum with the received value $r_i^{\{i,j\}}$

$$r_i^{\{i,j\}} \oplus r_j^{\{i,j\}} = (s_{ui} \otimes s_{vj}) \oplus (s_{uj} \otimes s_{vi})$$

(Proved in [29] section 7.5.2.2.)

So P_i , after getting all $r_j^{\{i,j\}}$-s from all parties with $j > i$, can compute its share for the output wire w.

$$s_{wi} = (s_{ui} \otimes s_{vi}) \oplus (\oplus_{i<j} ((s_{ui} \otimes s_{vj}) \oplus (s_{uj} \otimes s_{vi}))$$

Given the methods of share based evaluation of both type gates, each party computes the circuits' output value shares. The computation of the full value of an output wire can be done after each participant sends the result of its computations to others.

Note, that computation of an XOR gate share values is free in terms of network communication and almost free, in terms of computation (n XOR operations). However, the computation of AND gates takes $\binom{n}{2}$ OT invocations.

3.2.3 Optimizations

3.2.3.1 White-Box oblivious transfer

Oblivious transfer is an essential cryptographic primitive heavily used in secure computations. Yao's garbled circuit based two-party protocol [12], GMW protocol considered in this paper and others also. It was introduced by Rabin in [13] as a protocol where the sender sends a message to receiver with $\frac{1}{2}$ probability and does not reveal if the receiver got the message or not. A modified version of this protocol was introduced later by Even et al. in [14], currently known as the basic 1-out-of-2 OT protocol. This protocol assumes that the receiver has a selection bit $s\epsilon\{0,1\}$ and the sender has two bits $m_0, m_1 \epsilon \{0,1\}$. After protocol execution the receiver gets m_s and nothing about m_{1-s} and the sender does not reveal s.

Unfortunately this protocol depends on public-key operations, which require exponentiation operations which are pretty expensive, considering the huge amount of OT operations required for secure computations(in Yao's protocol OTs are used for all input bits of one of the participants, in GMW OTs are used for all AND gates). In [11] was introduced a new approach to OT using white-box cryptography techniques instead of public-key operations, allowing to reduce cost of an OT in several orders of magnitude, in case of many invocations, as reported.

3.2.3.2 WBOT extension

Another optimization technique for reducing the OT invocation count in secure multiparty computation frameworks is OT extension introduced by Beaver in [74] and later enhanced by [78]. Beaver also defined a method for preprocessing oblivious transfers preliminarily to use later with help of relatively cheap operations [34, 35]. We have constructed similar optimization

for white-box OT, enabling the reduction of the required OT invocation count to fixed security parameter. The details of the extension can be found in [52].

3.2.3.3 Low-depth circuit construction

In secure computation protocols based on binary circuits the count of gates is of a great importance and directly affects the execution time. In GC protocol for the two-party setting by virtue of optimization techniques (free-XOR [41]) and GMW protocol for multi-party secure computations by definition computation of XOR gates takes little effort and the evaluation efficiency of AND gates affects the overall computation greatly.

In GMW protocol unlike the GC two-party protocol, the count of AND gates is not the only parameter affecting the performance. Depth of the circuit being evaluated has crucial effect on performance also: the deeper is the circuit– the worse is performance for circuit evaluation of same depth. For overcoming this issue we added special circuit building constructions with slightly larger overall gate count but less depth specially for lowering circuit overall circuit depth. Some details of those circuit constructions can be viewed in Table 1.

We use several replacements for extensively used functionalities like addition, squaring and comparison.

Computing the addition

The most common method used for addition of l-bit numbers is the Ripple-carry adder ADD_{RC}^{l}, overall gate count and depth of which are linear with respect to l: $S(ADD_{RC}^{l}) = D(ADD_{RC}^{l}) = l$ [42]. Although the overall size of the adder is considered minimal for this case, we are more concerned with the depth property of the circuit, because in our implementation of the GMW protocol we can compute many and gates in parallel, but the parallel executions are limited to the gates on the same level in circuit i.e. not

depending on each other. For those purposes we prefer using circuit blocks with larger size, but lower depth.

Table 1: depth and overall size of circuit building blocks [8]

Circuit	Size **S**	Depth **D**
Addition		
Ripple-carry ADD/SUB$^{\ell}_{RC}$	ℓ	ℓ
Ladner-Fischer ADD$^{\ell}_{LF}$	$1.25\ell\lceil\log_2 \ell\rceil + \ell$	$2\lceil\log_2 \ell\rceil + 1$
LF subtraction SUB$^{\ell}_{LF}$	$1.25\ell\lceil\log_2 \ell\rceil + 2\ell$	$2\lceil\log_2 \ell\rceil + 2$
Carry-save ADD$^{(\ell,3)}_{CSA}$	$\ell + \mathbf{S}(\mathrm{ADD}^{\ell})$	$\mathbf{D}(\mathrm{ADD}^{\ell})+1$
RC network ADD$^{(\ell,n)}_{RC}$	$\ell n - \ell + n - \lceil\log_2 n\rceil - 1$	$\lceil\log_2 n - 1\rceil + \ell$
CSA network ADD$^{(\ell,n)}_{CSA}$	$\ell n - 2\ell + n - \lceil\log_2 n\rceil$ $+\mathbf{S}(\mathrm{ADD}^{\ell+\lceil\log_2 n\rceil}_{LF})$	$\lceil\log_2 n - 1\rceil$ $+\mathbf{D}(\mathrm{ADD}^{\ell+\lceil\log_2 n\rceil}_{LF})$
Multiplication		
RCN school method MUL$^{\ell}_{RC}$	$2\ell^2 - \ell$	$2\ell - 1$
CSN school method MUL$^{\ell}_{CSN}$	$2\ell^2 + 1.25\ell\lceil\log_2 \ell\rceil - \ell + 2$	$3\lceil\log_2 \ell\rceil + 4$
RC squaring SQR$^{\ell}_{RC}$	$\ell^2 - \ell$	$2\ell - 3$
LF squaring SQR$^{\ell}_{LF}$	$\ell^2 + 1.25\ell\lceil\log_2 \ell\rceil - 1.5\ell - 2$	$3\lceil\log_2 \ell\rceil + 3$
Comparison		
Equality EQ$^{\ell}$	$\ell - 1$	$\lceil\log_2 \ell\rceil$
Sequential greater than GT$^{\ell}_{S}$	ℓ	ℓ
D&C greater than GT$^{\ell}_{DC}$	$3\ell - \lceil\log_2 \ell\rceil - 2$	$\lceil\log_2 \ell\rceil + 1$
Selection		
Multiplexer MUX$^{\ell}$	ℓ	1
Minimum MIN$^{(\ell,n)}$	$(n-1)(\mathbf{S}(\mathrm{GT}^{\ell})+\ell)$	$\lceil\log_2 n\rceil(\mathbf{D}(\mathrm{GT}^{\ell})+1)$
Minimum index MIN$^{(\ell,n)}_{IDX}$	$(n-1)(\mathbf{S}(\mathrm{GT}^{\ell})+\ell+\lceil\log_2 n\rceil)$	$\lceil\log_2 n\rceil(\mathbf{D}(\mathrm{GT}^{\ell})+1)$
Set Operations		
Set union $\cup^{\ell}$	ℓ	1
Set intersection $\cap^{\ell}$	ℓ	1
Set inclusion $\subseteq^{\ell}$	$2\ell - 1$	$\lceil\log_2 \ell\rceil + 1$
Count		
Full Adder count CNT$^{\ell}_{FA}$	$2\ell - \lceil\log_2 \ell\rceil - 2$	$\lceil\log_2 \ell\rceil$
Boyar-Peralta count CNT$^{\ell}_{BP}$	$\ell - d_H(\ell)$	$\lfloor\log_2 \ell\rfloor$
Distances		
Manhattan distance DST$^{\ell}_{M}$	$2\mathbf{S}(\mathrm{SUB}^{\ell})+\mathbf{S}(\mathrm{ADD}^{(\ell,3)})+1$	$\mathbf{D}(\mathrm{SUB}^{\ell})+\mathbf{D}(\mathrm{ADD}^{(\ell,3)})+1$
Euclidean distance DST$^{\ell}_{E}$	$2\mathbf{S}(\mathrm{SUB}^{\ell})+2\mathbf{S}(\mathrm{SQR}^{\ell})$ $+\mathbf{S}(\mathrm{ADD}^{(2\ell,4)})+2\mathbf{S}(\mathrm{MUX}^{\ell})$	$\mathbf{D}(\mathrm{SUB}^{\ell})$ $+\mathbf{D}(\mathrm{SQR}^{\ell})+3$

Ladner-Fischer Adder

The Ladner-Fischer adder ADD^{l}_{LF} [56] also referred as parallel prefix has logarithmic depth and implements the addition of two l-bit values x_l and y_l.

The adder has size $S(ADD_{LF}^{l}) = 1.25l\lceil \log_2 l \rceil + l$ and depth $D(ADD_{LF}^{l}) = 2\lceil \log_2 l \rceil + 1$. The speciality of Ladner-Fischer adder is the evaluation of several carry bits in simultaneously. A bit $p_{i,j}$ for parity and a carry bit $c_{i,j}$ are computed for all nodes positions varying $1 \leq i \leq l$ and levels varying $0 \leq j \leq \lceil \log_2 l \rceil$.

Parity and carry bits are computed by the adder in the zero level with gates

$$p_{i,0} = x_i \oplus y_i \text{ and } c_{i,0} = x_i \otimes y_i.$$

For subsequent levels, those bits are represented in the following way:

$$p_{i,j} = p_{i,j-1} \otimes p_{k,j-1} \text{ and } c_{i,j} = (p_{i,j-1} \otimes c_{k,j-1}) \otimes c_{i,j}.$$

Where k is the node which is responsible for propagation of the carrying bit to position i.

Finally, the adder computes the resulting sum in the following way at the level number $\lceil \log_2 l \rceil + 1$:

$$s_{l+1} = c_{l,\lceil \log_2 l \rceil}, \; s_i = p_{i,0} \oplus c_{i-1,\lceil \log_2 (i-1) \rceil}$$

$$\text{for } 1 < i \leq l, \text{ and } s_1 = p_{1,0}.$$

Squaring an integer

A number can be squared with multiplying it with itself, so a circuit construction for multiplication can be used for implementing this operation. However a fast parallel squarer construction SQR_L^l was presented by Yoo et al. which has better size properties compared to the multiplication based circuit. In this approach the improvement is made using the common squaring

by multiplication $\sum_{i=0}^{l} 2^{i-1}(x^l x_i)$ with use of carry-save and Ladner-Fischer adders [58].

$$S(SQR_L^l) = l^2 + 1.25l\lceil \log_2 l \rceil - 1.5l - 2$$

$$D(SQR_L^l) = D(ADD_{CSA}^{2l,\lceil \frac{l}{2} \rceil} + D(ADD_{LF}^{2l}) + 1 = 3\lceil \log_2 l \rceil + 3.$$

Comparison

Two comparing operators are used in our circuit constructions which are equality check operator and greater-than operator. The standard equality circuit has $l - 1$ bit comparisons, so has linear size [41]. It can be modified to play-off manner circuit to reduce its depth to logarithmic complexity.

Greater-than operator checks if its first operand is graeter than the second, so it can be implemented like in [42] with linear size and depth. However logarithmic complexity for depth can be achieved her also with the help of greater-than circuit GT_{DC}^l constructed in [57] with

$S(GT_{DC}) = 3l - \lceil \log_2 l \rceil - 2$ and depth $D(GT_{DC}^l) = \lceil \log_2 l \rceil + 1$.

3.2.4 Experimental results

In this section some experimental results are presented displaying improvements of our protocol compared to Choi et al. implementation of GMW protocol [9]. This implementation was the best available GMW based approach to multiparty computation.

We made comparison of several results of the same function's Boolean circuit generated by our compiler. The function evaluated is basically a secure peer rating system, which allows several users to rate each other by some factors

and find out only the final results of the rating, without revealing the individual ratings to each other or other entities.

The experiments were made in a network of 10 devices with from 4 to 8 GB of RAM and 4th generation Intel Core i5 CPUs. The computers were connected via 1Gb Ethernet. In Fig. 19 we show the comparison of results on circuit with 5, 10, 50 and 100 participants in our implementation versus GMW protocol implementation by Choi et al. with the OT extension of the latter disabled, to demonstrate the improvement of our framework against the traditional GMW. The improvement in its significant part is the result of using white-box cryptography based oblivious transfer protocol in its extended version described in chapter 2.

In Fig. 20 we compare our results with the ones got on the full optimized implementation of Choi et al.. Here we also find significant improvement which is explained by the use of WBOT and its extension.

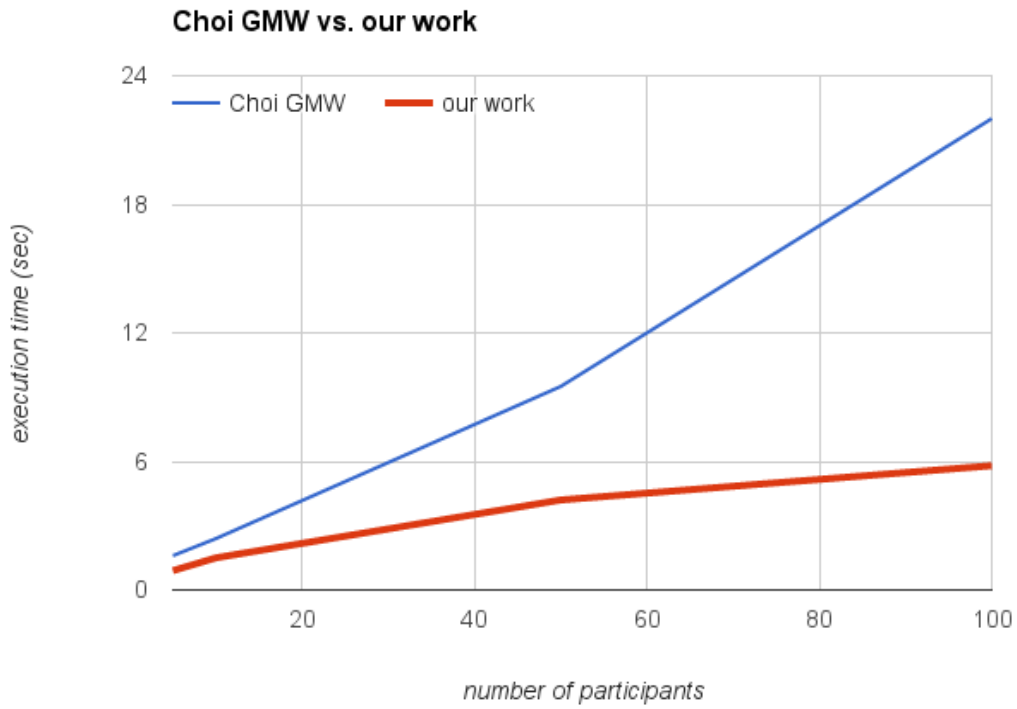

Fig. 19: Choi et al. un-extended vs. our implementation

We computed the execution time for both platforms in a similar way. While the individual execution times for each participant can be computed easily, the overall performance of the platform can be assessed in several ways. For the deployment of our platform we used shared locations for keeping non-secret information. The Boolean circuit outputted by our compiler is shared among the participants along with a file containing information about the state of users (state file).

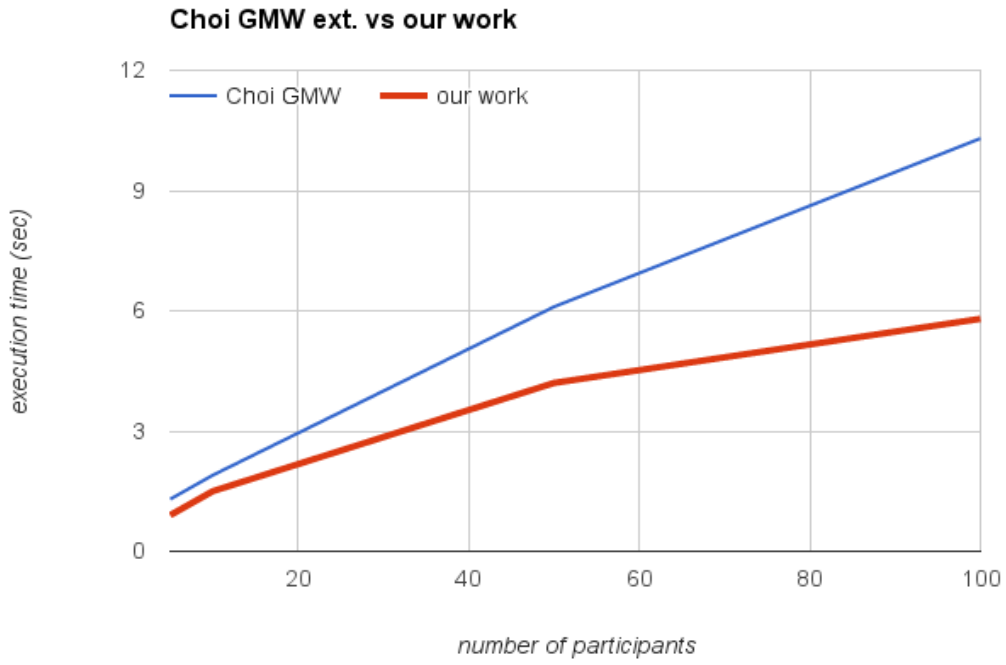

Fig. 20: Choi et al. vs. our implementation. Both extended

Each participant registers its IP and corresponding communication port, after it's ready for the computation (has the Boolean circuit along with generated secret shares for its inputs). We take the starting point of execution time the edit of the state file after which all parties are marked as ready. As the finish of the execution we take the timestamp of the state file edit, after which all parties are marked as finished. An interesting problem to consider is avoiding all shared means for completely decentralizing the execution of our platform even in terms of non-secret information.

3.3 Extension to GMW

Here we introduce a secure multiparty computation protocol intended for computations including active and passive participants. The active parties have private input and expect output from the computation, as regular participants described in the context of GMW protocol. Passive participants have their private inputs to the function being computed and also expect output, but do not want to participate in computation. This protocol is motivated by the scenarios, where several parties cooperate to compute a function while keeping the privacy of their inputs, but one or more of the parties are not willing to be directly involved in the evaluation processes due to lack of computational resources, low bandwidth connection, limited networking capabilities or other reasons. The passive participants might use mobile devices, which nowadays usually do not lack computational power, but have limited battery life which is being heavily used during wireless network communications.

Having implemented this protocol we can also assume having a tool for secure outsourcing application via constructing a special platform on this protocol, with only one party of the first type is allowed e.g. has input, but does not want to participate in computation (further we will refer to those as *non-computing parties*) and the others is of the second type e. g. has no input to the function being computed.

The proposed protocol is based on GMW SMC protocol with an alternative approach to secret sharing [7]. Here the secrets are shared between parties willing to partake in the computation (further we will refer to those as *computing parties*) only i.e. if there are overall $n + m$ participants in the computation with only n computing parties, only n secret shares would be generated for any argument, which would be shared between computing parties only.

We will assume that not all of the computing parties are being engaged in adversarial behaviour, because as non-computing parties do not hold shares for their own inputs, their secrets might be recovered by computing parties if those are maliciously cooperating.

3.3.1 Collaboration between competing services

Here we will describe the process of combining multiple delivery services into a unified platform that involves several members. In fact the application described is suitable for using in different contexts like combining any delivery or transportation services where the members are service providers, who do not want to disclose their current locations or the locations they are available to cover within fixed time to their competitors, but eager to cooperate with them to have their share in unified ordering system, thus increasing their order counts and service efficiency.

Concerning the motivation of client to use the unified system, let's compare the actions needed for getting the fastest service. In case of having an access to the unified application, the client just has to give the application her location and confirm the order. All the computation and decision processes are completed without the clients' involvement. In absence of such a system one should contact several service providers (possibly with different interfaces), give them her location, compare the offers of different providers, find the best of them and finally put an order. In the latter case the client also has to trust the information she got from service providers. Suppose the service providers are taxi service companies with one or more cabs and the client has no priorities for choosing particular service other than fast pick-up.

Taxi services do not want to make their locations visible to rival companies to avoid them getting advantage by better positioning of their cabs (for example, this can be done by a company with significantly more cabs). All cab locations

are also hidden from the client, because a competitor can possibly act as a client to find out others' cab locations and get advantage.

Having the application scenario described we show several methods as candidates for the secure computation itself. These methods are basically computing the minimum value of distances between client as one point, and provider instances as candidates for the second point. Different distance measurement algorithms are presented below.

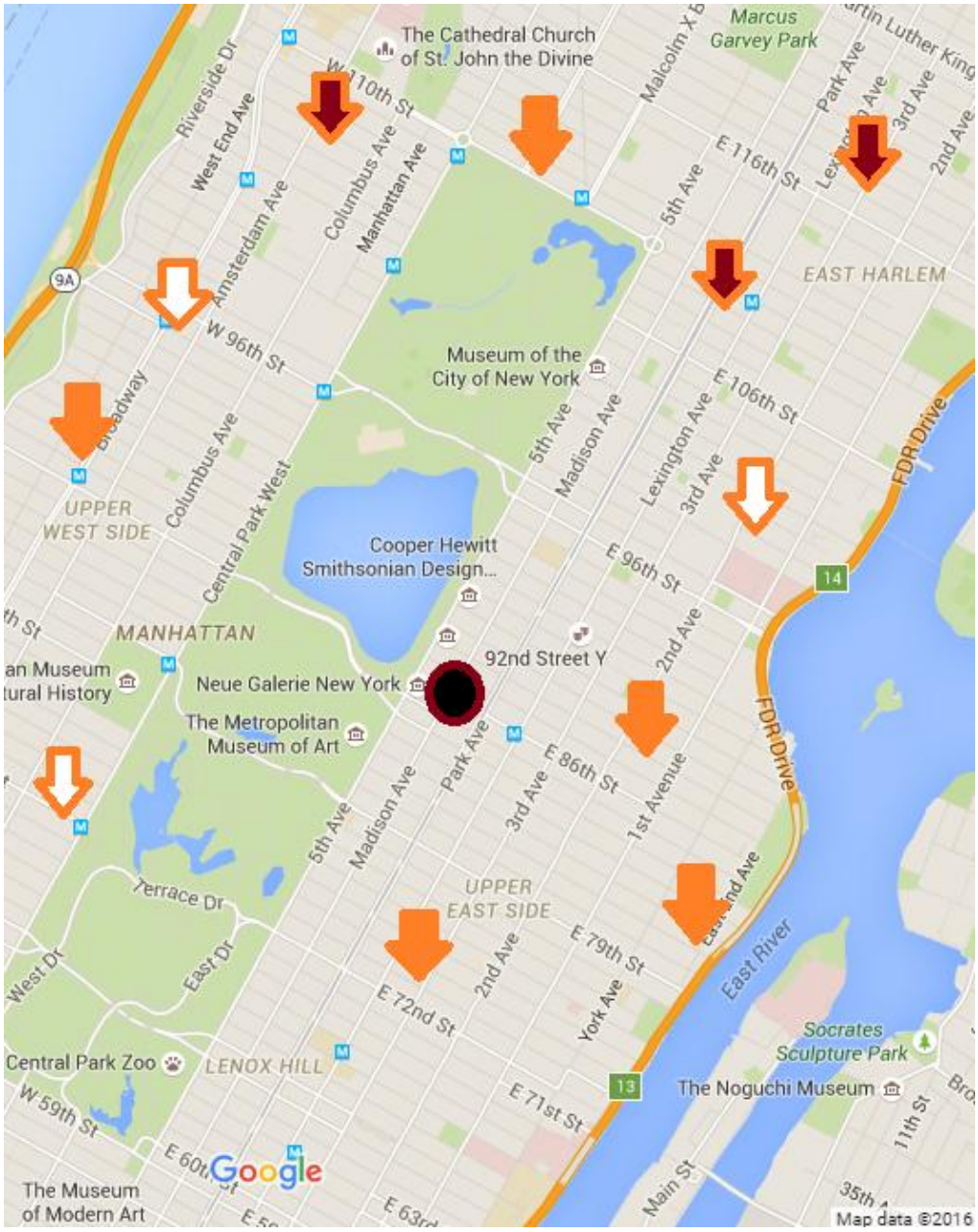

Fig. 21: Taxi service car locations and client location

3.3.1.1 Distance Measurement

The main purpose of this application is to find the nearest item in the unified database of locations to a submitted location. With the computation securing method already appointed, we still need to have accurate location detecting and network connection hardware for the service providers and individual cabs. In this work we consider the mentioned hardware implemented in a black-box manner – we do not the technical and efficiency details of those. We also assume that the computing parties also have identical maps and their locations follow common coordinate system, to avoid misinterpretation of function arguments.

Considering the distance measurement we have several methods as candidate. For Euclidean and Manhattan distances Depth efficient circuits were used from [8].

3.3.1.2 Euclid Distance

The most general version of the distance measurement function is computation of the Euclid distance between the customer and all provider instances (e.g. cabs). With the coordinates given in two dimensional format, the customer's location denoted as (x_c, y_c) and for i-th provider instance (x_i, y_i) considering m provider instances, the following function should be computed to determine the "winning" bid:

$$\min_{i<m}((x_c - x_i)^2 + (y_c - y_i)^2).$$

Although, this distance measurement method will be very precise in case of services operating in seas and oceans or drone deliveries, it's accuracy can be

very poor in case of taxi or delivery services in most cities street maps or for similar problems in any grid-like structures.

3.3.1.3 Manhattan Distance

Fig. 22: Manhattan (dashed) versus Euclid distance (solid)

This distance measurement algorithm is also called taxicab distance which intuitively fills the accuracy gap mentioned for Euclid distance measurement.

The distance in this method is the sum of absolute differences of the Cartesian coordinates. In this case the desired function is:

$$\min_{i<m}(|x_c - x_i| + |y_c - y_i|).$$

This algorithm is very simple and has a small circuit representation and, therefore, is efficient in terms of performance, but unfortunately it cannot be considered as a general purpose method.

3.3.1.4 Graph Approach.

Another method for fixed traffic map but with rather irregular structure can be introduced with graph construction for specific maps, where grid-oriented Manhattan approach is not effective. This technique assumes the parties have once preprocessed the map they are going to cooperate upon and generated a graph-map for it. The generated graph should be an oriented and weighted. The graph vertices are crossroads on the map and an edge (A, B) is present in the graph between vertices A and B, if there's a direct route without crossroads (not involving any other vertex) connecting those on the real map with traffic allowed in $A \rightarrow B$ direction. The weights are non-negative numbers assigned to the graph edges, equal to the distance between corresponding crossroads.

Upon our function computation we can also add special marked vertices for the client and provider instances or simply mark their nearest nodes as special, depending on the map specifics.

With this graph construction our desired function to compute the nearest provider instance will be with use of Dijkstra's algorithm [10] for finding shortest path in graphs. The algorithm will terminate upon finding any node marked as special.

Conclusion

In this thesis the design and implementation of a generic framework for secure multiparty computation were described, along with construction of its building blocks, including an extension protocol to white-box oblivious transfer protocol which decreases the required invocation count of transfers, resulting in enhancement of the overall performance of the applications based on the framework. The lifecycle of implemented framework consists of the following stages:

- The participants of computation input a program implementing their desired function described in a high level programming language.
- A single pass Boolean circuit implementing the same function is being generated by the compiler module of the framework.
- The generated circuit is being executed securely based on GMW secure multiparty computation protocol.

The implemented framework consists of separately developed compiler and computation module, which are of general purpose and can be used in other frameworks or be replaced in future with enhanced versions easily. The underlying oblivious transfer protocol is also replaceable with other implementations for satisfying stronger security requirements than in the semi-honest model or for achieving better performance. The current implementation displays increased performance compared to existing implementations. The advantage was mainly achieved by virtue of applying a novel oblivious transfer protocol based on white-box cryptography in the extended version as a building block in the implementation of Goldreich-Micali-Widgerson (GMW) protocol for secure multiparty computations offering security in semi-honest adversarial model.

A compiler, which generates Boolean circuits optimized for GMW protocol for any function defined in special high level language is included in the framework, enabling users without expert knowledge in Boolean circuit construction to interact with the framework effectively.

Also an extended version of the GMW protocol was introduced, which enables participation of computationally passive entities in the process by only providing inputs to the function and getting output of the protocol execution.

Applications scenarios for traditional and extended versions of the implemented protocol utilizing the framework were suggested and justified.

Table of Figures

References

[1] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In STOC '87: Proceedings of the nineteenth annual ACM symposium on Theory of computing, pages 218–229, New York, NY, USA, 1987. ACM.

[2] K.G. Paterson; G.J. Watson. "Immunising CBC Mode Against Padding Oracle Attacks: A Formal Security Treatment". Security and Cryptography for Networks – SCN 2008, Lecture Notes in Computer Science (Springer Verlag), pp. 340–357, 2008.

[3] S. Vaudenay, "Security Flaws Induced by CBC Padding Applications to SSL, IPSEC, WTLS...". Advances in Cryptology – EUROCRYPT 2002, Proc. International Conference on the Theory and Applications of Cryptographic Techniques (Springer Verlag), pp. 534–545, 2002.

[4] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for noncryptographic fault-tolerant dist ributed computations. In 20th Annual ACM Symposium on Theory of Computing, pages 1–10. ACM Press, 1988.

[5] D. Chaum, C. Crépeau, and I. Damgård. Multiparty unconditionally secure protocols. In 20th Annual ACM Symposium on Theory of Computing, pages 11–19. ACM Press, 1988

[6] R. Cramer, I. Damgård, and J. Nielsen. Multiparty computation from threshold homomorphic encryption. In Birgit Pfitzmann, editor, Advances in Cryptology – EUROCRYPT 2001, vol. 2045, pp. 280–300, 2001.

[7] A. Shamir. "How to share a secret." Communications of the ACM 22, no. 11, pp.612-613, 1979.

[8] T. Schneider and M. Zohner. "GMW vs. Yao? Efficient secure two-party computation with low depth circuits." In Financial Cryptography and Data Security, pp. 275-292. Springer Berlin Heidelberg, 2013.

[9] S.G. Choi, K-W. Hwang, J. Katz, T. Malkin, D. Rubenstein. "Secure multiparty computation of Boolean circuits with applications to privacy in on-line marketplaces." Dunkelman, O. (ed.) CT-RSA 2012. LNCS, vol. 7178, pp. 416–432. Springer, Heidelberg, 2012.

[10] M. O. Rabin, "How to Exchange Secrets by Oblivious Transfer", Technical Memo TR-81, Aiken Computation Laboratory, 1981

[11] M. O. Rabin. Digitalized Signatures and Public-Key Functions as Intractable as Factorization, MIT/LCS/TR-212, 1979.

[12] S. Even, O. Goldreich and A. Lempel. "A Randomized Protocol for Signing Contracts." Communications of the ACM, 28(6):637{647, 1985.

[13] M. Fischer, S. Micali and C. Rackoff. "A secure protocol for oblivious transfer", presented at Eurocrypt'84 but missing in the proceedings.

[14] J. Kilian. "Founding Cryptography on Oblivious Transfer". In 20th STOC, pages 20-31, 1988.

[15] B. Chor, E. Kushilevitz, O. Goldreic and M. Sudan. "Private information retrieval." Journal of the ACM (JACM) 45, no. 6, 965-981, 1998.

[16] S. Wiesner, "Conjugate coding," SIGACT News, vol. 15, no. 1, pp. 78–88, 1983.

[17] Rivest, R.; Shamir, A.; Adleman, L.. "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems". Communications of the ACM 21 (2): 120–126, 1978.

[18] C. Gentry and Z. Ramzan. "Single-database private information retrieval with constant communication rate." In Automata, Languages and Programming, pp. 803-815. Springer Berlin Heidelberg, 2005.

[19] M. Naor and B. Pinkas. "Efficient oblivious transfer protocols." In Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms, pp. 448-457, 2001.

[20] A. Jivanyan and G. Khachatryan “Efficient Oblivious Transfer Protocols based on White-Box Cryptography”. AUA Internal Reports, 2013.

[21] Y. Lindell and B. Pinkas. "Secure multiparty computation for privacy-preserving data mining." Journal of Privacy and Confidentiality 1, no. 1, 2009.

[22] M. Bellare and S. Micali. "Non-interactive oblivious transfer and applications." In Advances in Cryptology—CRYPTO'89 Proceedings, pp. 547-557. Springer New York, 1989.

[23] E. Kushilevitz, and R. Ostrovsky. "Replication is not needed: Single database, computationally-private information retrieval." In focs, pp. 364. IEEE, 1997.

[24] G. Brassard, C. Crépeau and J.-M. Robert “All-or-Nothing Disclosure of Secrets”. Proc. Advances in Cryptology - Crypto ’86, Springr-Verlag LNCS 263 (1987), 234– 238

[25] D. Boneh, “The Decision Diffie-Hellman Problem”, Proc. of the Third Algorithmic Number Theory Symposium, Springer-Verlag LNCS 1423 (1998) 48–63.

[26] W. Diffie and M. Hellman, New directions in cryptography, IEEE Trans. Inform. Theory, vol. 22(6), 1976, 644–654.

[27] T. ElGamal, “A public key cryptosystem and a signature scheme based on discrete logarithms”, Proc. Advances in Cryptology – Crypto ’84, Springr-Verlag LNCS 196 (1985), 10–18.

[28] O. Goldreich, “Foundations of Cryptography: Volume 1, Basic Tools”, Cambridge University Press, 2007.

[29] O. Goldreich, “Foundations of Cryptography: Volume 2, Basic Applications”, Cambridge University Press, 2009.

[30] A. Yao, “How to Generate and Exchange Secrets”, 27th Annual Symposium on Foundations of Computer Science - Toronto, 1986.

[31] V. Kolesnikov and R. Kumaresan, “Improved Secure Two-Party Computation via Information-Theoretic Garbled Circuits”, Security and Cryptography for Networks - 8th International Conference, 2012.

[32] A. Shelat and C.-H. Shen, “Fast two-party secure computation with minimal assumptions”, 2013 ACM SIGSAC Conference on Computer and Communications Security- 2013.

[33] J. Massey, G. Khachatrian and M. Kuregian, “Nomination of SAFER+ as a Candidate Algorithm for Advanced Encryption Standard (AES),” represented at the first AES conference, Ventura, USA, August 20-25, 1998.

[34] D. Beaver. “Precomputing oblivious transfer.” In Advances in Cryptology | Crypto 95, volume 963 of LNCS, pp. 97-109. Springer, 1995.

[35] Bogetoft P., Christensen D. L., Damgård I., Geisler M. et al. “Secure multiparty computation goes live”, Financial Cryptography and Data Security, Springer - 2009, p. 325-343.

[36] G. Di Crescenzo, “Private selective payment protocols”, Financial Cryptography - 2001.

[37] D. Malkhi, N. Nisan, B. Pinkas and Y. Sella, “Fairplay - Secure Two-Party Computation System”, Proceedings of the 13th USENIX Security Symposium - August 9-13, 2004.

[38] SFDL Specification - Version 2.0 - Septempber 4, 2008. [Online].Available:http://www.cs.huji.ac.il/project/Fairplay/FairplayMP/SFDL2.0.pdf.

[39] Y. Huang, D. Evans, J. Katz and L. Malka, “Faster Secure Two-Party Computation Using Garbled Circuits”, USENIX Security Symposium - 2011.

[40] C. E. Shannon, :A symbolic analysis of relay and switching circuits”, American Institute of Electrical Engineers, Transactions of the - 1938 - vol. 57, no. 12 - p. 713-723.

[41] V. Kolesnikov and T. Schneider, “Improved Garbled Circuit: Free XOR Gates and Applications”, Automata, Languages and Programming, 35th International Colloquium - 2008.

[42] V. Kolesnikov, A.-R. Sadeghi, and T. Schneider. "Improved garbled circuit building blocks and applications to auctions and computing minima." In Cryptology and Network Security, pp. 1-20. Springer Berlin Heidelberg, 2009.

[43] C.E. Shannon. “A mathematical theory of communication”. The Bell System Technical Journal, 27:379–423, 623–, july, october 1948

[44] T. Kerins and K. Kursawe. “A cautionary note on weak implementations of block ciphers”. In 1st Benelux Workshop on Information and System Security (WISSec 2006), page 12, Antwerp, BE, 2006.

[45] A. Shamir and N. van Someren. “Playing “Hide and Seek” with Stored Keys”. In Proceedings of the Third International Conference on Financial Cryptography (FC 1999), volume 1648 of Lecture Notes in Computer Science, pages 118–124. Springer-Verlag, 1999

[46] B. Schneier. “Description of a New Variable-Length Key, 64-bit Block Cipher (Blowfish)”. In Proceedings of the International Workshop on Fast Software Encryption (FSE 1993), volume 809 of Lecture Notes in Computer Science, pages 191–204, London, UK, 1994. Springer-Verlag.

[47] W. Michiels and P. Gorissen. “Cryptographic Method for a White-Box Implementation". U.S. Patent Application 2010/0080395 A1, Filed November 9, 2007.

[48] D. Genkin, A. Shamir and E. Tromer. "RSA key extraction via low-bandwidth acoustic cryptanalysis." In Advances in Cryptology–CRYPTO 2014, pp. 444-461. Springer Berlin Heidelberg, 2014.

[49] S. Goldwasser and S. Micali. "Probabilistic encryption & 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

[50] A. Jivanyan, G.H. Khachatryan, T. V. Sokhakyan and D. H. Danoyan. “Acceleration of Secure Function Evaluation Protocol”. Computer Science and Information Technologies (CSIT), 2015, pp 115-118.

[51] D. H. Danoyan and T. V. Sokhakyan, “A Generic Framework for Secure Computations”. Proceedings of the Russian-Armenian (Slavonic) University #2, Physical-Mathematical Sciences, 2015, pp. 14-21.

[52] D. H. Danoyan, “Extending White-Box Cryptography Based Oblivious Transfer Protocol”. Proceedings of the Yerevan State University #1 (239), Physical and Mathematical Sciences,2016, pp. 40-44.

[53] D. H. Danoyan, “Secure Multiparty Computations for Collaboration Between Competing Services”. Mathematical Problems of Computer Science #2, 2016.

[54] A. Ben-David, N. Nisan, and B. Pinkas. “FairplayMP: A system for secure multi-party computation”. In 15th ACM Conf. on Computer and Communications Security, pp. 257-266. ACM Press, 2008.

[55] T. V. Sokhakyan, “A user configurable compiler for secure computation framework” In Proceedings of Engineering Academy of Armenia, Vol. 13, N. 1, 2016, pp. 138-142.

[56] R. Ladner and M. Fischer, "Parallel prefix computation." Journal of the ACM (JACM) 27, no. 4 (1980): 831-838.

[57] J. Garay, B. Schoenmakers and J. Villegas. "Practical and secure solutions for integer comparison." In Public Key Cryptography–PKC 2007, pp. 330-342. Springer Berlin Heidelberg, 2007.

[58] J. Yoo, K.F. Smith, and G. Gopalakrishnan. "A fast parallel squarer based on divide-and-conquer." Solid-State Circuits, IEEE Journal of 32, no. 6 (1997): 909-912.

[59] T. K. Frederiksen, T. P. Jakobsen, J. B. Nielsen, P. S. Nordholt and C. Orlandi, "MiniLEGO: Efficient Secure Two-Party Computation from General Assumptions.", EUROCRYPT, 2013.

[60] Y. Huang, J. Katz and D. Evans, "Quid-Pro-Quo-tocols: Strengthening Semi-honest Protocols with Dual Execution," IEEE Symposium on Security and Privacy, SP 2012, 21-23 May, 2012.

[61] Y. Lindell, B. Pinkas and N. P. Smart, "Implementing two-party computation efficiently with security against malicious adversaries," in Security and Cryptography for Networks, Springer, 2008, pp. 2-20.

[62] D. Evans, Y. Huang, J. Katz and L. Malka, "Efficient privacy-preserving biometric identification," in Proceedings of the 17th conference Network and Distributed System Security Symposium, NDSS, 2011.

[63] D. Boneh, R.A. DeMillo, and R.J. Lipton. "On the importance of checking cryptographic protocols for faults." In Advances in Cryptology—EUROCRYPT'97, pp. 37-51. Springer Berlin Heidelberg, 1997.

[64] P. Kocher, J. Jaffe, and B. Jun. Differential Power Analysis. In "Advances in Cryptology – CRYPTO '99", Lecture Notes in Computer Science (1999), 388–397.

[65] S. Chow, P. Eisen, H. Johnson, and P.C. van Oorschot. White-Box Cryptography and an AES Implementation. In "Selected Areas in Cryptography: 9th Annual International Workshop, SAC 2002", Lecture Notes in Computer Science (2003), 250–270.

[66] S. Chow, P. Eisen, H. Johnson, and P.C. van Oorschot. A White-box DES Implementation for DRM Applications. In “Digital Rights Management: ACM CCS-9 Workshop, DRM 2002”, Lecture Notes in Computer Science (2003), 1–15.

[67] O. Billet, H. Gilbert, and C. Ech-Chatbi. Cryptanalysis of a White Box AES Implementation. In “Selected Areas in Cryptography: 11th International Workshop, SAC 2004”, Lecture Notes in Computer Science (2005), 227–240.

[68] H. Vollmer, “Introduction to Circuit Complexity,” Berlin, Springer, 1999.

[69] E. Allender, M. Loui, and K. Regan, “Complexity classes,” In M. Atallah, editor, “Algorithms and Theory of Computation Handbook”, ch. 27, CRC Press, 1999.

[70] W. Stallings. “Cryptography and Network Security”, 4th Edition. Pearson Education India, 2006.

[71] https://en.wikipedia.org/

[72] N. Ferguson and B. Schneier. Practical cryptography. Vol. 23. New York: Wiley, 2003.

[73] H. Lipmaa. "First CPIR protocol with data-dependent computation." In Information, Security and Cryptology–ICISC 2009, pp. 193-210. Springer Berlin Heidelberg, 2009.

[74] D. Beaver. Correlated Pseudorandomness and the Complexity of Private Computations, STOC 1996: 479-488

[75] P. Mohassel and M. Franklin. "Efficiency tradeoffs for malicious two-party computation." In Public Key Cryptography-PKC 2006, pp. 458-473. Springer Berlin Heidelberg, 2006.

[76] Y. Aumann and Y. Lindell. Security Against Covert Adversaries: Efficient Protocols for Realistic Adversaries. In 4th TCC, Springer-Verlag (LNCS 4392), pages 137-156, 2007.

[77] C. Hazay and Y. Lindell. "Efficient protocols for set intersection and pattern matching with security against malicious and covert adversaries." In Theory of Cryptography, pp. 155-175., 2008.

[78] Y. Ishai, J. Kilian, K. Nissim and E. Petrank. "Extending oblivious transfers efficiently." In Advances in Cryptology-CRYPTO 2003, pp. 145-161. Springer Berlin Heidelberg, 2003.

[79] G. Asharov, Y. Lindell, T. Schneider, and M. Zohner. "More efficient oblivious transfer and extensions for faster secure computation." In Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security, pp. 535-548. ACM, 2013.

[80] G. Asharov, Y. Lindell, T. Schneider, and M. Zohner. "More efficient oblivious transfer extensions with security for malicious adversaries." In Advances in Cryptology--EUROCRYPT 2015, pp. 673-701. Springer, 2015.

[81] ISO/IEC. (2014). ISO International Standard ISO/IEC 14882:2014(E) – Programming Language C++. International Organization for Standardization (ISO). Retrieved from https://isocpp.org/std/the-standard.

[82] M. Osadchy, B. Pinkas, A. Jarrous and B. Moskovich. "Scifi-a system for secure face identification." In Security and Privacy (SP), 2010 IEEE Symposium on, pp. 239-254. IEEE, 2010.

[83] A. Miyaji and M.S. Rahman, “Privacy-preserving data mining in presence of covert adversaries”. In Advanced Data Mining and Applications (pp. 429-440). Springer Berlin Heidelberg, 2010.

[84] H. Carter, C. Amrutkar, I. Dacosta and P. Traynor, “For your phone only: custom protocols for efficient secure function evaluation on mobile

devices," Journal of Security and Communication Networks (SCN), vol. 7, no. 7, pp. 1165-1176, 2014.

[85] R. Rivest, L. Adleman and M. Dertouzos, "On data banks and privacy homomorphisms," in Proceedings of the IEEE Annual Symposium on Foundations of Computer Science, 1978.

[86] C. Gentry, "Fully homomorphic encryption using ideal lattices," In Proceedings of the Annual ACM Symposium on Theory of Computing, 2009.

[87] Z. Brakerski and V. Vaikuntanathan, "Fully homomorphic encryption from ring-lwe and security for key dependent messages," in CRYPTO, 2011.

[88] R. Cramer, I. Damgård and J. Nielsen, "Multiparty computation from threshold homomorphic encryption," in EUROCRYPT, 2001.

[89] I. Damgård, M. Geisler, M. Krøigaard, and J. Nielsen. "Asynchronous multiparty computation: Theory and implementation". In 12th Intl. Conference on Theory and Practice of Public Key Cryptography - PKC 2009, volume 5443 of LNCS, pp. 160-179. Springer, 2009.

[90] S. Goldwasser, Y. Kalai, R. Popa, Vaikuntanathan V. and N. Zeldovich, "Succinct functional encryption and applications: Reusable garbled circuits and beyond," in Proceedings of the Annual ACM Symposium on Theory of Computing, 2013.

[91] R. Gennaro, C. Gentry and B. Parno, "Non-interactive veriable computation: Outsourcing computation to untrusted workers," in CRYPTO, 2010.

[92] S. Gorbunov, V. Vaikuntanathanand D. Wichs, "Leveled fully homomorphic signatures from standard lattices," in Proceedings of the Annual ACM Symposium on Theory of Computing, 2015.