Implementation detailsRelated links

Contact us

Picture

**Secret Sharing**

# Secure Multi Party Computation project

## The following website describes a project that was designed and written as a part of a workshop in computer security in the Tel-Aviv university.

## Workshop instructor : Prof. Ran Canetti.

## Workshop assistant : Rani Hod.

## Workshop website : Security workshop spring 09.

## Project submmiters : Doron Friedland, Yael Smith, Bar Katz and Erez

**Alon.**

**General info:**

**Secure multi-party computation is a cryptographic paradigm that allows a set of mutually distrustful parties to perform some joint computation in a way that guarantees correctness of the outputs along with secrecy of local data (to the maximum possible extent).**

**The *BenOr-Goldwasser-Wigderson (BGW)* work from STOC 1988 ,"*Completeness theorems for non-cryptographic fault-tolerant distributed computation*", describes the following:**

**"A protocol is *t-private* if any set of at most $t$ players cannot compute after the protocol more than they could jointly compute solely from their set of private inputs and outputs.**
**A protocol is *t-resilient* if no set of $t$ or less players can influence the correctness of the outputs of the remaining players. For this to make sense, the function definition should be extended to specify what it is if some players neglect to give their inputs or are caught cheating.**

**Theorem 1: For every (probabilistic) function $f$ and $t < n/2$ there exists a *t-private* prolocol.**

**Theorem 2: There are functions for which there we no *n/2-private* protocols.**

**Theorem 3: For every probabilistic function and every $t < n/3$ there exists a protocol that is both *t-resilient* and *t-private*.**

**Theorem 4: There are functions for which there is no $n/3$-*resilient* protocol. "**

**In our project we have implemented a MPC protocol which is both $t$-*private* and $t$-*resilient* for $t < n/3$.**

**We have also implemented a MPC protocol which is only $t$-*private* for $t < n/2$.**

**The users can choose the protocol that they would like to use before the computation begins.**

**Project description:**

**Our goal was to implement a specific multi-party computation protocol. The given protocol was written in an abstract format, with many details missing.**

**The implementation needed to include both *regular* and *byzantine* cases of the protocol and to fill out the missing details.**

**The implemented protocol is the protocol that is described in the *BenOr-Goldwasser-Wigderson (BGW)* work from STOC 1988 - "*Completeness theorems for non-cryptographic fault-tolerant distributed computation*" (link to MIT's document).**

**Originally, due to performance condiserations, the used multiplication protocol was chosen to be the one that is described in the *Gennaro-***

*M.Rabin-T.Rabin* (GRR) work from PODC 1998 - "*Simplified VSS and fast-track multiparty computations with applications to threshold cryptography*" (link to IBM's document).

As part of the project we have been asked to develop a compiler that gets a user written function in a simple form as an input and generates a computation circuit from it. This computation circuit is composed of addition and multiplication gates.

After the computation circuit is generated, the MPC protocol can run over it and calculate the desired results.

Secret sharing in a nutshell:

This section is meant to give the reader (you) a taste of what secret sharing really is.

Don't be afraid of the notations, it is quite simple.

Lets say some player wants to share some secret $S$ with some other $n-1$ players ($n$ players over all).

He wants each of the players to have a part of his secret but he doesn't want any player to be able to conclude anything about his secret (it is an important secret).

In order to share his secret in a protected way, the player can do the following:

- Choose some fixed finite field E such that $|E| > n$ and $S$ is an element within E.

- **Each player Pi {i = 0,..., *n-1*} should be associated with a number *bi* from the field E.**

- **Select *t* random elements ai from E, i = 1,...,*t*. The selected random elements will define the following polynomial from *t* degree :**

  **f(x) = S + a1·x + a2·x^2 + ... + at·x^t.**

- **Now, for each player Pi the player should calculate and send the value *Si* = f(*bi*). That Si is the part of the secret given to player Pi.**

**As in Shamir's secret sharing scheme, the sequence (*S0, S1,...,Sn-1*) is a sequence of *t*-wise independent random variables uniformly distributed over E. Thus the value of the secret is completely independent from the shares {*Si*}.**

**Notice that the free coefficient of the polynomial f(x) contains the secret and f(0) = *S*.**

**In addition, in order to discover the secret the players will have to interpolate the polynomial f(x) and then calculate f(x).**

**Because f(x) is from *t* degree, this can be done only if there are *t* players**

**who share their shares between them.**

**As you may noticed, to minimize the chance that *t* players will try to discover the secret it is recommended to choose a higher *t* degree.**

**In order to be able to interpolate the polynomial f(x) upon demand, *t***

must be smaller or equal to $n$.

The multiplication protocol:

The multiplication protocol is a main component of the computation circuit.

In general, a multiplication gate is more complicated than an addition gate because it doubles the degree of the calculation polynomial. This raise in the polynomial degree requires the secret shares to be randomized and re-shared by each player upon each multiplication.

Projects achievements:

Our project was logically divided into two main parts,

*Regular case MPC protocol* , a $t$ -private protocol.

*Byzantine case MPC protocol*, a $t$ -private and $t$ -resilient protocol.

## Regular case MPC protocol abilities:

- Matrices and regular numbers manipulations over finite fields.
- Secret Sharing and recombination according to Shamir Secret Sharing scheme.
- BGW and GRR multiplication step*.
- Compiler that gets an input in a certain format and generate a computation circuit from it

computation circuit from it.

- Basic communication abilities between players.
- Using XML file for Loading players' communication details.

- User friendly interface.
- Graphical representation of the computation circuit.

* We are using the GRR algorithm in the multiplication step because it requires less communication steps.

## Byzantine case MPC protocol abilities:

- Absolute verification of a secret.
- Exclude byzantine players from the calculation process.
- Welch-Berlekamp error correcting code algorithm for BCH codes.
- Together with the project's instructor, Prof. Ran Canetti, we have developed a solution for the multiplication step.
- Advanced communication abilities between players - developing the ability to send any objects between players.
- Adding the possiblity to load players' communication details from a server.
- Enhanced graphical user interface that signals deception attempts to the user.

## Implementation details:

**The project was implemented in Java using NetBeans as an IDE.**

**Our analysis and design have divided the project into four main modules:**

- **MPC protocol module.**
- **Compiler module.**
- **Connection controller module.**
- **GUI module.**

**Each module is extendable and logically separated from the other**

**modules.**

**This design allowed us to develop the separated modules in parallel.**

**In addition to the mentioned implemented capabilities, we have implemented byzantine players**

**in order to verify that our implementation handles the protocol's different scenarios.**

**Please follow the link to the workshop website, there you can find:**

- **A documentation with elaborated description of the implementation and the different modules.**

- **The project's source code.**

## Project main difficulty

**After thoroughly studying the GRR's multiplication protocol we have found out that the suggested solution for the byzantine case is incomplete.**

**After consulting with the project's instructor, Prof. Ran Canetti, we have decided that in the regular case we will use GRR's multiplication step and in the byzantine case we will use a solution that was developed together with Ran Canetti as a part of the project.**

## Project related links

- *"Completeness theorems for non-cryptographic fault-tolerant distributed computation", BenOr-Goldwasser-Wigderson* work from STOC 1988.
- "*Simplified VSS and fast-track multiparty computations with applications to threshold cryptography*", *Gennaro-M.Rabin-T.Rabin* (GRR) work from PODC 1998.
- "Algorithmic Introduction to Coding Theory" – Madhu Sudan (2001).
- Security workshop spring 2009 website.
- The Fairplay project.
- The VIFF project, Virtual Ideal Functionality Framework.
- Secure multi party computation on wikipedia.

## Contact details

## Contact details

**Project email address: project.mpc@gmail.com.**

**Spring 2009 Tel Aviv University**

Established by Erez Alon