# Unser erstes Dokument

Latex Tutorial

08. Februar 2013

## Inhaltsverzeichnis

# 1 Einleitung

## 1.1 Zielsetzung

Das Ziel der Arbeit ist es, die Neuerungen in dem Paper [1] zu implementieren. Einerseits, um zu demonstrieren, dass die vorgestellten Protokolle und Funktionalitäten sicher und effizient sind. Andererseits, um die neuen Funktionalitäten und ?Verschnellerungen? allen zugänglich zu machen.
Um die Funktionalitäten wie im Paper [1] beschrieben zu implementieren, werden auch Implementierungen von anderen Veröffentlichungen benötigt ([**?**] ). Da zu diesem Zeitpunkt noch keine derartigen Implementierungen veröffentlicht sind, kann auch nicht auf diese Zurückgegriffen werden. Um trotzdem zu demonstrieren, dass das im [1] vorgestellte Protokoll funktioniert und effizient ist, habe ich die Unterprotokolle, die auf solche externen Paper zurückgreifen auf unsichere Weise implementiert. Dadurch funktioniert das Protokoll, und die unsicheren Unterprotokolle können ohne viel Zeitaufwand durch sichere Implementierungen ersetzt werden, wenn die Funktionalitäten der anderen Veröffentlichungen implementiert sind.

# 2 Grundlagen

## 2.1 Homomorphe Verschlüsselungen

Homomorphe Verschlüsselung ist eine Form der Verschlüsselung, die bestimmte Berechnungen auf verschlüsselten Daten erlauben, und dann ein verschlüsseltes Ergebnis zurückgibt. Wenn dieses dann wieder entschlüsselt wird, ist das Ergebnis das gleiche, wie wenn diese Arbeitsschritte auf dem unverschlüsselten Anfangswert ausgeführt worden wären. [2]
In dieser Arbeit wird im speziellen additiv homomorphe Verschlüsselung benutzt. Diese erlaubt das addieren von zwei verschlüsselten Zahlen und das multiplizieren einer verschlüsselten mit einer unverschlüsselten Zahl.
Diese Funktionalitäten sind für das Protokoll wichtig, denn in jedem Unterprotokoll wird mit verschlüsselten Daten gerechnet. Und nur so können wir trotz einer sicheren Verschlüsselung mit den Daten rechnen, ohne den Inhalt der Verschlüsselung zu kennen.

# 3 Multiparty Cardinality Testing

## 3.1 Neues am Paper

## 3.2 Abwandlungen zum Paper

Das Protokoll wurde offensichtlich entworfen um auch mit mehreren teilnehmenden Parteien effizient zu sein. Das Paper [1] nutzt also in vielen Unterprotokollen Broadcast-Funktionen.

BEISPIEL: secRank, auch OMM, auch MPCT

Ich habe versucht, nah an die Spezifizierungen des Papers ran zu kommen, doch um die Implementierung des Protokolls zu vereinfachen und um das Testen der Implementierung zu erleichtern, habe ich das System etwas abgewandelt. In meiner Implementierung gibt es einen Koordinator. Dieser Koordinator erleichtert das testen enorm, denn durch ihn kann sicher gestellt werden, dass alle "verschicktenÏnformationen immer zum richtigen Zeitpunkt am richtigen Ziel ankommen.
Diese Änderung wird die Sicherheit des Protokolls nicht schwächen, weil alle Informationen, per Broadcast verschickt werden immer verschlüsselt sind. Auch wenn Daten entschlüsselt werden, sind sie immer für einen Angreifer, der Informationen über die Eingabemengen erhalten will nutzlos. Ein gutes Beispiel dafür ist das Unterprotokoll OInv
HIER BEISPIEL EINFÜGEN
Hier wird zwar eine Matrix entschlüsselt, diese ist aber randomisiert, also nutzlos, außer, man besitzt eine der geheimen Matrizen, die ja nicht verschickt werden.

Das Protokoll ist also auf eine Art und Weise aufgebaut, dass kein passiver Zuhörer irgendwelche nützlichen Informationen aus den Nachrichten, die zwischen den Parteien verschickt werden, gewinnen kann. Also kann auch ein Angreifer, der alle Nachrichten zwischen den Teilnehmern des Protokolls keine Informationen extrahieren. Also können wir auch sicher sein, dass, selbst wenn der Koordinator von einem Angreifer kontrolliert wird, die verschlüsselten Eingabedaten sicher sind.
Die neue Struktur, die einfacher zu implementieren und testen ist, macht das Protokoll also nicht weniger sicher. Besonders deshalb, weil das Protokoll so konstruiert ist, dass selbst alle verschickten Nachrichten zusammen keine geheimen Daten preisgeben.

### 3.3 andere Paper

# 4 Funktion der Software

## 4.1 Verschlüsselungen

#### 4.1.1 DDH/Damgard-Jurik

#### 4.1.2 Übertragung der Python Implementierung

## 4.2 Architektur

## 4.3 Ring Realisierung

Da in vielen der Unterprotokolle (OLS, ODT, OMM, secRank, OInv, SUR) Berechnungen über endlichen Körpern stattfinden, um Korrektheit und Sicherheit zu gewährleisten, benötigen wir eine robuste und effiziente Möglichkeit, mit endlichen Körpern zu arbeiten.
Da wir unter anderem lineare Gleichungssysteme über endlichen Körpern lösen müssen, und es nur wenige Java-Bibliotheken gibt, die diese Funktionalität anbieten, ist die Open-Source Bibliothek JLinalg [?] gut für unseren Fall geeignet. Die Bibliothek

bietet aber standardmäßig nur eingeschränkte Funktionalitäten für beliebige Körper an. Das Lösen von linearen Gleichungssystemen ist eines der Dinge, die aber nur in den Rationalen Zahlen oder dem Körper {0;1} möglich sind. Daher war es augenscheinlich am einfachsten, die Bibliothek zu erweitern, indem ich eine neue Klasse erstellt habe, wodurch die Bibliothek jetzt auch andere Funktionalitäten über beliebig großen Körpern berechnen kann. Unter anderem kann so auch das lineare Gleichungssystem gelöst werden.

AUSSCHNITT DER NEUEN KLASSE

### 4.4 sichere Matrix Berechnungen

## 5 Sicherheit der Software

## 6 Probleme der Software

### 6.1 Nicht alle Protokolle sicher implementiert

### 6.2 Bei großen Zahlen interferieren von Ring/Verschlüsselung

## 7 Fazit

## 8 Literatur

## Literatur

[1] Nico Döttling. Multiparty cardinality testing for threshold private setintersection. 2021.

[2] X Yi, R Paulet, and E Bertino. Homomorphic encryption. in: Homomorphic encryption and applications. springerbriefs in computer science, 2014.