Aus dem Institut für Technische Informatik der Universität zu Lübeck
Direktor: Prof. Dr. rer. nat. Rüdiger Reischuk

Privately computing the intersection of two SNP sets

Berechnung des Schnitts zweier SNP Mengen unter Erhalt der Privatsphäre

Praktikumsbericht
im Rahmen des Studienganges Medizinische Informatik
der Universität zu Lübeck

vorgelegt von
Niklas Jobst

ausgegeben und betreut von
Prof. Dr. rer. nat. Rüdiger Reischuk

mit Unterstützung von
Florian Thaeter

Lübeck, den May 23, 2018

Abstract

Ziel dieses Praktikums war es zu erörtern, wie zwei Parteien die Ähnlichkeit ihrer DNA berechnen können, ohne, dass dabei eine der Parteien Informationen über den genetischen Code der jeweils anderen erlangt.

Die Grundlagen für diese Berechnungen basieren auf bereits existierenden Methoden, mit welchen der Schnitt zweier Mengen unter Sicherung der Privatsphäre berechnet werden kann.

Im Zuge dieses Praktikums habe ich drei dieser Methoden mit Bezug zum gegebenen Anwendungsfall implementiert und deren Effizienz miteinander vergleichen:

- R.Egert et al. : Privately Computing Set-Union and Set-Intersection Cardinality via Bloom Filters, LNCS volume 9144, 2015
- A.Davidson et al. : An Efficient Toolkit for Computing Private Set Operations, LNCS volume 10343, 2017
- S. K.Debnath et al. : Secure and Efficient Private Set Intersection Cardinality Using Bloom Filter, LNCS volume 9290, 2015

Contents

1 Einleitung

1.1 Next generation sequencing

In den ersten Jahren des 21.Jahrhunderts wurden mehrere Sequenzierungstechnicken entwickelt, welche unter dem Begriff next generation sequencing (NGS) zusammengefasst werden. Sie alle vereint die Fähigkeit, DNA in großem Maßstab parallel sequenzieren zu können. Dies hatte zur Folge, dass NGS Sequenzierungen im Vergleich zur klassischen Sanger Sequenzierungen in den darauf folgenden Jahren deutlich schneller und günstiger wurden. Kostete die Sequenzierung eines Genoms 2000 noch über 10 Millionen US Dollar, sind die Preise bis zum heutigen Tag auf unter 1000 gefallen.

1.2 Genetisch marker

In diesem Projekt wird die DNA der beiden Parteien als Mengen betrachtet. Aufgrund der Tatsache, dass der Großteil der DNA bei allen Menschen identisch ist, nutzte ich genetische Marker, welche die DNA unterscheiden. Der Schnitt dieser beiden Marker dient dann als Maß der Ähnlichkeit der jeweiligen DNAs.

1.2.1 Genetische Marker

Unter genetischen Markern werden bestimmte klar definierte Sequenzen und Positionen im genetischen Code können dazu genutzt werden Personen zu identifizieren.

SNPs

INDELs

1.2.2 Personalisierte Medizin

In der personalisierte Medizin werden individuelle Eigenschaften von Personen berücksichtigt die

1.3 Anwendung

1.3.1 Personalisierte Medizin

In der personalisierte Medizin werden individuelle Eigenschaften von Personen berücksichtigt, insbesondere genetische In der Personalisierten Medizin sind Therapien bestimmte genetische Profile gekoppelt. Um festzustellen, ob eine Therapie für einen Patienten zulässig ist, muss daher zunächst sein genetischer Code mit dem für diese Therapie notwendigem verglichen werden. Derzeit werden diese Vergleiche ohne die entsprechenden Datensicherheits-Vorkehrungen vorgenommen. Ziel dieses Praktikums war es durch Anwendung der genannten Methoden die Sicherung der Privatsphäre bei der Durchführung eines solchen Vergleichs zu erhöhen.

2 Methoden

Im Zuge dieses Praktikums habe ich zwei Methoden

2.1 Bloom Filter

Alle diese Methoden basieren auf sogenannten Bloomfiltern. Hierbei handelt es sich um eine Technik um festzustellen, ob bestimmte Daten in einem Datensatz vorhanden sind oder nicht. Sie bestehen aus einem mit Nullen vorinitialisiertem m Bit langen Array und k Hashfunktionen, welche auf die Positionen des Arrays abbilden.

Zur Initialisierung werden auf jedes Element des Datensatzes alle k Hashfunktionen angewendet. Die zur Ausgabe der Hashfunktionen korrespondierenden Bits im Array werden darauf hin auf Eins gesetzt.

Soll für ein Datenelement geprüft werden, ob dieses Teil des Datensatzes ist, werden alle Hashfunktionen auf dieses angewendet.
Nur wenn alle Positionen im Array an den korrespondierenden Punkten der Ausgabe dem Wert Eins entsprechen wird angenommen das sich das Element im Datensatz befindet.

Diese Überprüfung ist jedoch nicht resistent gegenüber Falsch Positiven Ergebnissen, da diese Positionen auch durch mehrere

2.2 Kryptosysteme

2.2.1 Homomorphie

Homomorphie bezeichnet eine Eigenschaft von Kryptosystemen. Ein Kryptosystem ist genau dann homomorph gegenüber einer mathematischen Operation, wenn Berechnungen im Ciphertext mit dieser Operation denen im KLartext entsprechen.

2.2.2 Elgamal

Bei Elgamal handelt es sich um ein im Jahr 1985 vom Kryptologen Taher Elgamal entwickeltes Public-Key-Verschlüsselungsverfahren. Elgamal ist eine Erweiterung des Diffie-Hellmann Schlüsselaustausches.

Verfahren

Zunächst wählt der Client eine endliche zyklische Gruppe Z der Ordnung q mit einem Generator g.

- Secrect key: Der Client wählt eine zufällige Zahl $a < q$ mit dem $GGT(a, q) = 1$. Dies ist der Secret key
- Public Key: Der public key ist dann $P = g^{\mathrm{a}}$

Sei $m \in Z_{\mathrm{q}}$ die zu versendende Nachricht. Dann wählt der Server eine zufällige Zahl $r < q$ mit dem $GGT(r, q) = 1$. Nun berechnet sich $c_1 = g^{\mathrm{r}}$ sowie $c_2 = P^{\mathrm{r}} * m$. Der Ciphertext besteht so aus $C = (c_1, c_2)$.

Zur Entschlüsselung wird $\Sigma = c_1^{-q} * c_2 brechnet$.

Homomorphie
Elgamal ist homomorph gegenüber der Multiplikation

$$E(m_1 * m_2) = (E(m_1) * E(m_2))$$

Sicherheit
Elgamal ist IND-CPA sicher, falls das

Dann gibt es einen Algorithmus, der in polyn. Zeit DH-Schlussel gab von zufalligen Gruppenelementen unterscheidet. Widerspruch: Nach Annahme gibt es keinen effizienten Algorithmus zum Entscheiden von DH-Schlusselngab. Daher kann es auch keinen polynomiellen Angreifer A geben.

2.2.3 Pailier

Verfahren

Schlüsselerzeugung:
Das Schlüsselpaar wird folgendermaßen generiert: Der Client wählt zwei Primzahlen p,q , mit ggt(pq, (p-1)(q-1))= 0. Des weiteren wird der Generator g so gewählt, sodass $g \in (\mathbb{Z}/n^2\mathbb{Z})$ und$n$ die Ordnung von g teilt. Das Schlüsselpaar wird dann folgendermaßen gebildet.

- Secrect key: $\lambda = kgV(p-1, q-1)$
- Public Key: (n, g)

Verschlüsselung:
Zur Verschlüsselung einer Nachricht $m \in \mathbb{Z}$ wählt der Client zunächst eine Zufalls Zahl r wobei $0 \leq r \leq n$
Dann berechnet sich der Ciphertext $c = g^m * r^n \mod n^2$
Entschlüsselung:
Der Plaintext kann folgendermaßen berechnet werden: $m = L(c^\lambda \mod n^2) * \mu mod n$

Homomorphie:
Paillier ist homomorph gegenüber der Addition.

$$E(m_1 + m_2) = (E(m_1) + E(m_2))$$

Sicherheit

2.2.4 Goldwasser-micali

Verfahren:

Schlüsselerzeugung:
Zunächst generiert der Client zwei Primzahlen p, q und berechnet dann $N = pq$. Des weiteren bestimmt er einen Nichtrest u und das Jakobi Symbol $(\frac{y}{n}) = 1$, sodass das Legendre Symbol $(\frac{y}{q} = \frac{y}{p} = -1)$

Das Schlüsselpaar wird dann folgendermaßen gebildet.

- Secrect key: (p, q)
- Public Key: (n, u)

Verschlüsselung:
Zur Verschlüsselung einer Nachricht $m \in (0, 1)$ wählt der Client zunächst eine Zufallszahl $r \in \mathbb{Z}$. Der ciphertext c berechnet sich dann aus: $c = y^m * u^2 mod n$ *Entschlüsselung:*
Zur Entschlüsselung X

Homomorphie:

Paillier ist homomorph gegenüber der Addition.

Sicherheit

2.3 Implementierte Algorithmen

2.3.1 Algorithmus 1 - Elgamal

Dieser Algorithmus wurde zunächst im ... veröffentlicht. Es wurden Algorithmen für unterschiedliche Konstellationen postuliert.

Der Client erstellt zu Beginn einen Boomfilter seiner Daten. Dabei wird jedes Datenelement einzeln Zur Verschlüsselung wählt der Client zunächst public und secret key nach elgamal. Daraufhin wird jedes Bit des Bloomfilter Arrays einzeln verschlüsselt. Hierzu werden die zu sendende Nachrichten so gewählt das $m = g^B F[i]$ Dies führt dazu, dass m an Stellen an welchen der Bloomfilter $BF_Client = 0$ dem Wert 1 entspricht und an den Stellen, an denen $BF_Client = 1$ dem Generator g.

$$S_\mathrm{i} = pk^{r_i} \ * \ \begin{cases} g^0 = 1 \ bei \ BF_1[i] = 1 \\ g^1 = g \ bei \ BF_1[i] = 0 \end{cases}$$

Diese Nachricht wird dann nach Elgamal verschlüsselt. Der Ciphertext entspricht dann:

$$(R_i, S_i) = (g^{\mathrm{r_i}}, pk^{\mathrm{r_i}} * g^{\text{1-BF}_1[\mathrm{i}]})$$

Der Ciphertext wird dann zusammen mit dem public key und den Bloomfilter Parametern an den Server übermittelt. Dieser erstellt nun seinerseits einen Bloomfilter mit den Einträgen seines Datensatzes unter Berücksichtigung der Parameter des Clients.

Für alle Indices an welchen $BF_Server = 0$ werden die Elemente des Ciphertextes des Clients R_i und S_i aufmultipliziert. Dies ist aufgrund der Homomorphie Eigenschaft von Elgamal ohne Datenverlust möglich.

Daraufhin selektiert dieser alle Indexes von Einträgen seines Bloomfilters. Indices Für jeden dieser Indices wird daraufhin der entsprechende Eintrag im Ciphertexts des Clients Aufmultiplikation von R_i bzw S_i an jenen Stellen, an welchen $BF_2 = 0$ ist.

$$V = (g^{s+r_{i_1}+r_{i_2}+ \ \dots \ +r_{i_k}})$$

$$W = \begin{cases} pk^{s+r_{i_1}+r_{i_2}+ \ \dots \ +r_{i_l}} * 1 \ falls \ BF_1 = 1, BF_2 = 0 \\ pk^{s+r_{i_1}+r_{i_2}+ \ \dots \ +r_{i_m}} * g^x \ falls \ BF_1 = BF_2 = 0 \end{cases}$$

Die Ergebnisse werden nun mit g^s bzw. pk^s rerandomisiert :

$$V = (g^\mathrm{s} * \Pi_{i:BF_2[i]=0} R_i)$$

$$W = (pk^\mathrm{s} * \Pi_{i:BF_2[i]=0} S_i)$$

V und W werden nun zurück an den Client gesendet, welcher nun die Elgamal Entschlüsselung auf diese anwendet:

$$\Sigma = W * V^{-sk}$$

Da $pk = g^{sk}$, ergibt sich folgende Gleichung:

$$\Sigma = (g^{sk*s+r_{i_1}+r_{i_2}+ \ ... \ +r_{i_k}} \ * \ g^{-sk*s+r_{i_1}+r_{i_2}+ \ ... \ +r_{i_k}} \ * \ g^z)$$

Nach dem Kürzen erhält man:

$$\Sigma = g^x$$

z entspricht hierbei der Anzahl an Positionen an denen sowohl der Client als auch der Server einen Nulleintag in ihren Bloomfiltern haben.

Der approximierte Betrag der im Bloomfilter gespeicherten Elemente errechnet sich dann durch:

$$|X| = \frac{ln(\frac{z}{m})}{k * \ ln(1 - \frac{1}{m})}$$

2.3.2 Algorithmus 2 - Paillier

Zunächst erstellt der Client einen Bloomfilter über seine Daten. Mittels des Pailier Kryptosystems wird wird jede Stelle des Bloomfilters einzeln verschlüsselt:

$$c_i = (g^{\mathrm{IBF[i]}} * {r_i}^{\mathrm{n}})$$

$$C_{\mathrm{i}} = r_i^n \ * \ \begin{cases} g^0 = 1 \ bei \ BF_1[i] = 1 \\ g^1 = g \ bei \ BF_1[i] = 0 \end{cases}$$

pk: public key, sk: private key, g: Generator r_i: Zufallszahlen aus Z_q

Der Server erstellt nun für jede Stelle einen eigen Bloomfilter. Daraufhin multipliziert er für jeden seiner Bloomfilter den Ciphertext des Clients an jenen Stellen auf, an welchen $BF_{server[j]} = 1$.

$$V_j = (g^{IBF_{i_1}+IBF_{i_2}+ \ ... \ +IBF_{i_k}} \ * \ r_{i_1}^n \ * \ r_{i_2}^n \ * \ ... \ * \ r_{i_k}^n)$$

$$V_j = \ r_{i_1}^n \ * \ r_{i_2}^n \ * \ ... \ * \ r_{i_k}^n \ \begin{cases} g^{1+1+1+ \ ... \ +1} \ wenn \ BF_c = 0, BF_{s[j]} = 1 \\ g^{0+0+0+ \ ... \ +0} \ wenn \ BF_c = BF_{s[j]} = 1 \end{cases}$$

$$\Sigma = W * V^{-sk}$$

V, W aus vorherigem Schritt einsetzen und für $pk = g^{sk}$

$$\Sigma = (g^{sk*s+r_{i_1}+r_{i_2}+ \ ... \ +r_{i_k}} \ * \ g^{-sk*s+r_{i_1}+r_{i_2}+ \ ... \ +r_{i_k}} \ * \ g^x)$$
$$\Sigma = g^x$$

The number of hash functions has distinct less influence

2.3.3 Algorithmus 3 - Goldwasser-Micali

3 Ergebnisse

Zum Test der Algorithmen habe ich verschiedene Mengen an Test SNPs erstellt, die sich zu unterschiedlichen Graden überschneiden.

Beide Algorithmen konnten selbst große Datensätze schnell vergleichen.

3.1 Algo elga

Überschneidung	14000	7500	5000	2000
Runtime (sec)	221	247	211	222
Abw. zur Überschn.	0.01%	3.3%	8.8%	36.8%

Table 3.1: Hashfunktionen : 14, Anzahl Bloomfilter Bits:3029660, Größe der Datensätze: 15000 SNPs

Array	1442696	1009887	577079	144270
Runtime (sec)	108	83	47	11
Abweichung	4%	6%	13%	51%

Table 3.2: Datensatz 1000 SNPs, Überschneidung 100, Hashfunktionen: 10

3.2 Algo 1

Array	14139	12119	10099	8080
Runtime (sec)	219	194	183	163
Abweichung	1%	4%	6%	24%

Table 3.3: Hashf.7, Überschneidung 100, SNPs 1000

4 Disskussion