Formel

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

$\Rightarrow$ Basiert auf Abschätzung der False-positive Wahrscheinlichkeit in einem Bloomfilter:

$\Rightarrow$ $z = m * (1 - \frac{1}{m})^{k* X}$

Algorithmus basierend auf Elgamal

Client

⇒ Erstellt Bloom Filter ihrer Daten

⇒ Verschlüsselt jede Stelle ihres Bloom Filters mittels ElGamal

$$(R_i, S_i) = (g^{r_i}, pk^{r_i} * g^{1\text{-}BF_1[i]})$$

$[pk, (R_i, S_i)]$ →

⇒ Alice entschlüsselt mit sk Ciphertext von Bob

⇒ Bestimmt Anzahl der Einträge an denen beide Bloom Filter null sind

⇒ Berechnet die Set-Union der BF

Server

⇒ Erstellt Bloom Filter seiner Daten

⇒ Selektiert jene Stellen Stellen in seinem BF die den Eintrag null besitzen.

⇒ Multipliziert an diesen Stellen die Werte des Ciphertextes von Alice auf

← (V, W)

⇒ Rerandomisiert die entstandenen Ergebnisse

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

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

Paillier - 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.

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

Algorithmus basierend auf Paillier

Client

⇒ Erstellt Bloom Filter ihrer Daten

⇒ Invertiert jede Stelle des Bloomfilters.

⇒ Verschlüsselt jede Stelle ihres Bloomfilters mittels Paillier

$$c = (g^m * r^n) \ mod \ n^2$$

⇒ Alice entschlüsselt mit sk Ciphertexte von Bob

⇒ Bestimmt Anzahl der Einträge an denen beide Bloom Filter null sind

⇒ Berechnet die Set-Union der BF

$[pk, c]$ →

← $rerand(c_{1-j})$

Server

⇒ Erstellt für jedes Element des Datensatzes einen Bloomfilter seiner Daten

⇒ Selektiert in jedem Blommfilter jene Stellen die den Eintrag Eins besitzen.

⇒ Addiert an diesen Stellen die Werte des Ciphertextes des Clients auf

⇒ Rerandomisiert die entstandenen Ergebnisse mit verschlüsselter Null

$$Rerand \ c_j = (cj * encrypt_{pailier}($$

Ergebnisse - Elgamal

⇒ Dauer für Vergleich des gesamten Exomes bei wenigen Minuten.

⇒ Laufzeit Unabhängig davon wie stark die Überschneidung zwischen zwischen den Datensätzen ist.

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

Table 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 2: Datensatz 1000 SNPs, Überschneidung 100, Hashfunktionen: 10

$\Rightarrow$ Die Laufzeit ist linear abhängig zur Anzahl der Bloomfilterbits

$\Rightarrow$ Die Stärke der Abweichung ist ebenfalls linear abhängig zur Anzahl der Bloomfilter Bits

Hashf.	1	4	7	10	14
Runtime (sec)	7	27	44	62	104
Abweichung	11%	13%	10%	9%	9%

Table 3: Datensatz 1000 SNPs, Überschneidung 100, Array: 504944

⇒ Die Laufzeit ist linear abhängig zur Anzahl der Bloomfilterbits

⇒ Die Stärke der Abweichung ist ebenfalls linear abhängig zur Anzahl der Bloomfilter Bits

Ergebnisse-Paillier

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

Table 4: Hashf.7, Überschneidung 100, SNPs 1000

⇒ Die Laufzeit ist linear abhängig zur Anzahl der Bloomfilterbits

⇒ Die Stärke der Abweichung ist ebenfalls linear abhängig zur Anzahl der Bloomfilter Bits