Websuche als Zero-Knowledge-Dienst

B. Sc. Phil Höfer

30. August 2019

Im aktuellen Diskurs über Websuchmaschinen ist Privatsphäre zu einem Kernthema geworden. Allerdings beschränkt sich dieses meist auf die Vertrauenswürdigkeit des Anbieters und die Frage, welche Aufzeichnungen dieser über die Suchen führt. In dieser kurzen Veröffentlichung wird auf die theoretischen langfristigen Möglichkeiten hingewiesen, die Privatsphäre einer Suche kryptografisch statt durch Vertrauen zu sichern.

1 Motivation

Mit zunehmender Digitalisierung und der Verlagerung vieler Rechtsgeschäfte in digitale Räume hat die Bedeutung der Computersicherheit stark zugenommen. Von andauernder Aktualität ist hierbei der Aspekt des Datenschutzes bzw. der Privatsphäre im Internet. Trotz anscheinender Sensibilisierung in der Zivilbevölkerung und empfindlicher Strafen für Unternehmen (vgl. EU-DSGVO) kommt es allerdings regelmäßig zu Sicherheitsmängeln und der Veröffentlichung eigentlich privater Datensätze.

Auch der Bereich der Internetsuchmaschinen ist von dieser Entwicklung betroffen: Der größte Teil der Suchanbieter wirbt, damit keine oder nur wenige persönliche Daten zu verarbeiten. Dabei ist das Ziel meistens, die *Vertrauenswürdigkeit* des Anbieters zu betonen, denn überprüfen lässt sich eine solche »No-Logging-Policy« von Seiten des Nutzers nicht. Demgegenüber steht ein Marktführerkonzern mit dem Versprechen, möglichst passende Suchergebnisse durch möglichst viele gesammelte Daten zu liefern.

Personalisierte, intelligente Suchergebnisse und eine privatsphärefreundliche Suche schließen einander aus – so scheint es, wenn man das Angebot der Suchmaschinenanbieter betrachtet. Doch in Zukunft mag dies nicht mehr stimmen, denn durch Fortschritte in der homomorphen Kryptografie kann es bald möglich sein, beide Aspekte zu vereinen und die Vertraulichkeit von Websuchen kryptografisch zu sichern.

Die homomorphe Kryptografie forscht daran, das Rechnen mit verschlüsselten Daten praktikabel zu machen. Es ist mittlerweile theoretisch möglich, beliebige Berechnungen mit Geheimtexten durchzuführen. Zur Zeit ist das allerdings noch mit unverhältnismäßigem Zusatzaufwand verbunden. Mit fortschreitender Forschung können aus den neuen kryptografischen Mitteln praktische Anwendungen hervorgehen.

Eine dieser Anwendungen, eine solchermaßen gesicherte Volltextsuchmaschine, wird im folgenden Text vorgestellt. Volltextsuchmaschinen sind elementare Werkzeuge, um große Datenbestände (z.B. das WWW) effizient zu durchdringen. Eine Web-Suchmaschine, die auf den vorgestellten kryptografischen Mechanismen basiert, kann trotz Personalisierung die Privatsphäre ihrer Nutzer erhalten und einen Zero-Knowledge-Ansatz für die Recherche im Web bieten.

2 Schaltnetze

Das vorgestellte Modell einer Suchmaschine basiert auf der Nutzung von Schaltnetzen, die hiermit kurz vorgestellt werden.

Definition 2.1. Ein **Schaltnetz** C ist ein Tupel (V, E, L, f_L), sodass folgendes gilt:

- (V, E) ist ein endlicher gerichteter azyklischer Graph aus einer Menge von Knoten V und einer Menge von Kanten E,
- L ist eine Menge von Booleschen Funktionen und
- f_L ist eine Funktion von V nach L, die jedem Knoten $v \in V$ mit Eingangsgrad i eine Boolesche Funktion $\lambda : \{0,1\}^i \to \{0,1\}$ aus L zuordnet.

Ein Knoten $v \in V$ eines Schaltnetzes (V, E, L, f_L) wird **Eingang** dieses Schaltnetzes genannt, wenn der Eingangsgrad von v gleich 0 ist. Ein Knoten $v \in V$ eines Schaltnetzes (V, E, L, f_L) wird **Ausgang** dieses Schaltnetzes genannt, wenn der Ausgangsgrad von v gleich 0 ist. Die Knoten eines Schaltnetzes werden auch als Gatter bezeichnet.

Metriken für Schaltnetze

Die **Größe** eines Schaltnetzes (V, E, L, f_L) ist die Zahl der Knoten, also die Anzahl der Elemente der Menge V. Die **Tiefe** eines Schaltnetzes (V, E, L, f_L) ist die Länge in Knoten des längsten einfachen Pfades im Graphen (V, E) von einem Eingang zu einem Ausgang des Schaltnetzes.

3 $\mathcal{C}$-Ausführungsschemas

Es folgt eine kurze Definition von Kryptosystemen in Anlehnung an Armknecht et al. [1] in einer Form, die für die spätere Vorstellung homomorph-kryptografischer Systeme gebraucht wird.

Definition 3.1. Für eine Menge $\mathcal{C}$ von Schaltnetzen ist ein **$\mathcal{C}$-Ausführungsschema** (engl. $\mathcal{C}$-evaluation scheme) ein Tupel $(\text{Gen}, \text{Enc}, \text{Eval}, \text{Dec})$ von Polynomialzeit-Algorithmen, sodass folgendes gilt:

- $\text{Gen}(1^\lambda)$ erzeugt aus einem Sicherheitsparameter λ ein Tupel $(\text{pk}, \text{sk}, \text{evk})$, bestehend jeweils aus einem öffentlichen und einem privatem Schlüssel – pk und sk – sowie einem Schlüssel evk, der für Berechnungen verwendet wird.
- $\text{Enc}(\text{pk}, m)$ ist ein Verschlüsselungs-Algorithmus. Er erzeugt aus einem öffentlichen Schlüssel pk und einem Klartext m einen Geheimtext c.
- $\text{Dec}(\text{sk}, c)$ ist ein Entschlüsselungs-Algorithmus zu Enc. Er erzeugt aus einem privaten Schlüssel sk und einem Geheimtext c einen Klartext m.
- $\text{Eval}(\text{evk}, C, c_1, \ldots, c_n)$ ist ein Algorithmus der ein Schaltnetz $C \in \mathcal{C}$ mithilfe des Schlüssels evk und Geheimtext-Eingaben c_1 bis c_n auswertet. Das Ergebnis ist ein neuer Geheimtext.

Definition 3.2. Ein $\mathcal{C}$-Ausführungsschema $(\text{Gen}, \text{Enc}, \text{Eval}, \text{Dec})$ verfügt über eine **korrekte Entschlüsselung**, wenn für alle Klartexte $m \in \mathcal{P}$ und die Ausgaben des Gen-Algorithmus sk, pk gilt:

$$\text{Dec}(\text{sk}, \text{Enc}(\text{pk}, m)) = m$$

Definition 3.3. Ein $\mathcal{C}$-Ausführungsschema $(\text{Gen}, \text{Enc}, \text{Eval}, \text{Dec})$ verfügt über eine **korrekte Ausführung**, wenn für alle Schaltnetze in $\mathcal{C}$ und alle Geheimtexte c_i aus dem Geheimtextraum $\mathcal{X}$ mit $\text{Dec}(\text{sk}, c_i) = m_i$ folgendes mit sehr großer Wahrscheinlichkeit gilt:

$$\text{Dec}(\text{sk}, \text{Eval}(\text{evk}, C, c_1, \ldots, c_n)) = C(m_1, \ldots, m_n)$$

Dabei sind sk und pk Ausgaben von $\text{Gen}(1^\lambda)$.

Definition 3.4. Ein $\mathcal{C}$-Ausführungsschema $(\text{Gen}, \text{Enc}, \text{Eval}, \text{Dec})$ ist **kompakt**, wenn es ein Polynom p abhängig von λ, dem Sicherheitsparameter der Gen-Funktion, gibt und die Ausgabelänge der Eval-Funktion nicht länger als $p(\lambda)$ ist.

4 Homomorphie

Unter einem Homomorphismus versteht man in der Mathematik eine Abbildung, bei der bestimmte (strukturelle) Eigenschaften nicht verändert werden:

$$f(a \circ_A b) = f(a) \circ_B f(b).$$

Ein anschauliches Beispiel für einen Homomorphismus ist die Exponentialfunktion. Die Multiplikation von zwei Instanzen der Exponentialfunktion mit der selben Basis entspricht einer Addition ihrer Exponenten:

$$e^x e^y = e^{x+y}$$

In diesem Zusammenhang bietet auch das RSA-Kryptosystem[13] in seiner Reinform Homomorphieeigenschaften, sofern keine Padding-Funktionen verwendet werden:

Im RSA-Verfahren ist die $Enc(pk, m)$-Funktion mit $pk = \left(\begin{smallmatrix} e \\ n \end{smallmatrix} \right)$ definiert als:

$$\text{Enc}(\left(\begin{smallmatrix} e \\ n \end{smallmatrix} \right), m) = m^e \mod n$$

Es ist offensichtlich, dass Multiplikationen homomorph trotz Verschlüsselung möglich sind:

$$\text{Enc}(pk, m_1) \cdot \text{Enc}(pk, m_2) = m_1^e \cdot m_2^e = (m_1 \cdot m_2)^e = \text{Enc}(pk, m_1 \cdot m_2) \mod n$$

RSA hat folglich homomorphe Eigenschaften im Bezug auf die Multiplikation.

5 Homomorphe Kryptosysteme

In homomorphen Kryptosystemen ist es möglich, Daten ohne vorherige Entschlüsselung für Berechnungen zu nutzen. Das heißt zum Beispiel, dass verschlüsselte – und somit unlesbare – Daten addiert, multipliziert oder mit anderen Operationen verwendet werden können. Im folgenden wird zwischen vollhomomorphen und teilhomomorphen Kryptosystemen (engl. *fully homomorphic* bzw. *somewhat homomorphic*) unterschieden. Während teilhomomorphe Systeme nur bestimmte Operationen wie Addition oder Multiplikation unterstützen, können vollhomomorphe Systeme beliebige Funktionen auf den verschlüsselten Daten ausführen. Offensichtlich sind vollhomomorphe Systeme nützlicher, jedoch sind teilhomomorphe Systeme deutlich einfacher zu realisieren. Eine zugängliche Vorstellung eines vollhomomorphen Systems findet sich in einem Web-Artikel von Lucas Barthelemy [3].

Definition homomorpher Ausführungsschemas

Definition 5.1. Ein $\mathcal{C}$-Ausführungsschema $(\text{Gen}, \text{Enc}, \text{Eval}, \text{Dec})$, das über eine korrekte Entschlüsselung und eine korrekte Ausführung verfügt, wird *teilhomomorph* (engl. *somewhat homomorphic*) genannt, wenn die $\mathcal{C}$ ungleich der leeren Menge ist.

Als Zwischenschritt zu vollhomomorphen Ausführungsschemas sollen auch geschichtet homomorphe Systeme kurz erwähnt werden. Diese sind eingeschränkte Varianten von teilhomomorphen Schemas, die nur Schaltnetze bis zu einer konstanten Tiefe ausführen können und folglich eine echte Teilmenge der teilhomomorphen Ausführungsschemas.

Definition 5.2. Ein kompaktes $\mathcal{C}$-Ausführungsschema $(\text{Gen}, \text{Enc}, \text{Eval}, \text{Dec})$, das zudem teilhomomorph ist, wird **geschichtet homomorph** (eng. *leveled homomorphic*) genannt, wenn die Gen-Funktion über einen zusätzlichen Parameter α verfügt, sodass das Schema nur Schaltnetze in $\mathcal{C}$ bis zu einer Tiefe von α ausführen kann und die Ausgabelänge der Dec-Funktion nicht von α abhängt.

Wenn ein geschichtet homomorphes Schema *alle* Schaltnetze bis zu der Tiefe α ausführen kann – die Menge $\mathcal{C}$ also der Menge aller Schaltnetze entspricht –, wird dieses als geschichtet vollhomomorph bezeichnet:

Definition 5.3. Für ein $\mathcal{C}$ gleich der Menge aller Schaltnetze wird ein geschichtet homomorphes $\mathcal{C}$-Ausführungsschema $(\text{Gen}, \text{Enc}, \text{Eval}, \text{Dec})$ auch als **geschichtet vollhomomorph** bezeichnet.

In einem geschichtet vollhomomorphen Ausführungsschema gibt es für jedes Schaltnetz C ein α, für das dieses Schaltnetz ausgeführt werden kann. Dieses α ist gleich der Tiefe von C. Ist die maximale Tiefe der ausgeführten Schaltnetze also bekannt, kann das α entsprechend gewählt werden.

Um daraus ein vollhomomorphes Schema zu erzeugen, hat Gentry[6] ein Verfahren namens *Bootstrapping* vorgestellt. Dabei wird die Nachricht entschlüsselt und neu verschlüsselt, ohne währenddessen im Klartext lesbar zu sein. Dafür werden die Enc- und Dec-Funktionen selbst in ein Schaltnetz umgewandelt, welches dann im geschichtet vollhomomorphen Schema ausgeführt wird. Dann braucht das Schema gerade groß genug sein, um eine kleine Menge an Operationen plus das eben erwähnte Umschlüsselungs-Netz ausführen zu können. Durch regelmäßiges *Auffrischen* des Geheimtextes können beliebige Schaltnetze ausgeführt werden.

Definition 5.4. Ein kompaktes $\mathcal{C}$-Ausführungsschema $(\text{Gen}, \text{Enc}, \text{Eval}, \text{Dec})$, das teilhomomorph ist, wird *vollhomomorph* genannt, wenn $\mathcal{C}$ gleich der Menge aller Schaltnetze ist.

6 Volltextsuche

Volltextsuche (engl. *full-text search*) bezeichnet im Wesentlichen das Durchsuchen aller Textinhalte einer Datenbasis. Ein *Dokument* ist ein Element einer Datenbank, das unabhängig von dieser Datenbank betrachtet werden kann. Jedes Dokument einer Datenbank verfügt über genau einen Selbstbezeichner, die sogenannte ID. Die *ID* eines Dokumentes ist ein textueller nicht-leerer Bezeichner in einem Dokument, der dieses Dokument innerhalb von dessen Datenbank eindeutig identifiziert. Der *Volltext* eines Dokumentes ist der gesamte in diesem Dokument enthaltene Text.

Definition 6.1. Eine **Volltextsuche** für eine Datenbank DB von n Dokumenten D_1 bis D_n ist ein Schaltnetz V_{DB} für das folgendes gilt:

- V_{DB} hat eine Anzahl an Eingängen, die gleich der Länge in Bit des längsten Wortes eines Dokumentes in der Datenbank ist.
- V_{DB} hat eine Anzahl an Ausgängen, die gleich der Länge in Bit der längsten ID eines Dokumentes in der Datenbank ist.
- Bei bitweiser Eingabe eines Wortes in das Schaltnetz wird an den Ausgängen die bitweise Repräsentation der ID eines Dokumentes erzeugt, das das eingegebene Wort in dessen Volltext enthält.
- Ist die Länge des Eingabewortes in Bit kleiner als die Anzahl der Eingänge, wird dieses durch Anhängen von 0-Bits verlängert.
- Ist ein Eingabewort im Volltext keines Dokumentes enthalten, wird alternativ an den Ausgängen das Wort erzeugt, das nur 0-Bits enthält.

7 Kryptografische Volltextsuche

Im Folgenden wird eine Implementierungsskizze geliefert, die die vorher definierte Volltextsuche durchführen kann. Es handelt sich dabei um eine möglichst anschauliche – und daher nicht vorrangig performante – Implementierung.

Ausgewählte bisherige Arbeiten

Çetin, Dai, Döroz und Sunar[5] reduzieren in ihrer Arbeit »Homomorphic Autocomplete« den Anwendungsfall der Volltextsuche auf eine automatische Wortvervollständigung, wie

sie einige Such-Dienste beherrschen. Sie entwickeln eine Implementierung auf GPUs, die sich unter bestimmten Bedingungen an die Praxistauglichkeit annähert.

Baldimitsi und Ohrimenko[2] hingegen haben sich in »Sorting and Searching Behind the Curtain« mit der Möglichkeit einer Ergebnissortierung beschäftigt. Sie entwickeln darin ein eigenes Schema, das Mehrwortsuchen mit sortierter Ausgabe ermöglicht.

8 Kryptografisches Suchkonzept

Aufbau der Suche

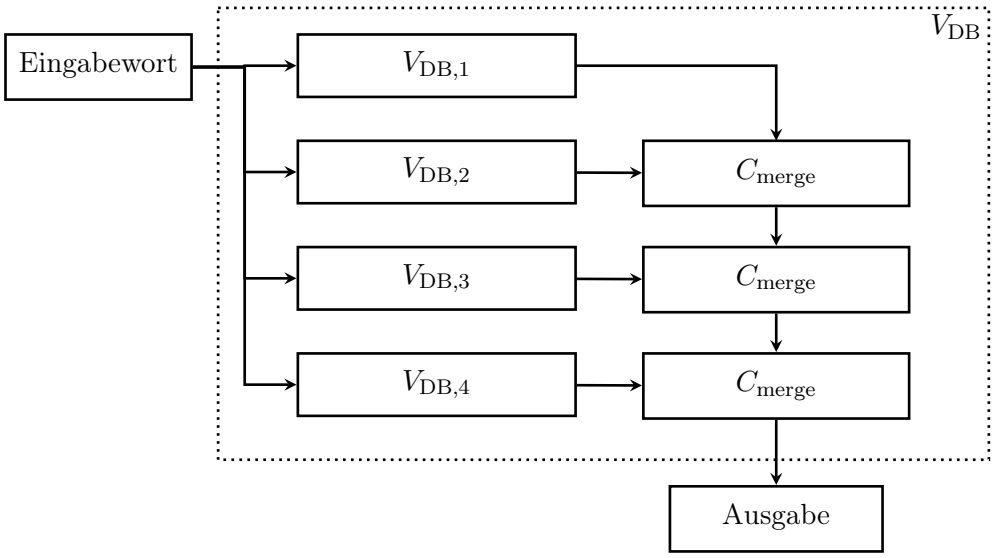

Abbildung 1: Übersicht der Implementierung

Die Volltextsuche V_{DB} verfügt über eine Menge von Eingängen, die der Länge in Bit des längsten zu erwartenden Suchwortes entspricht, und genug Ausgänge, um die ID eines beliebigen Dokumentes in der Datenbank ausgeben zu können. Die Anzahl der Eingänge und die Anzahl der Ausgänge, werden im folgenden mit m_{IN} bzw. m_{ID} bezeichnet.

Für jedes Dokument in der Datenbank wird ein Teilnetz erzeugt. Für eine Volltextsuche V_{DB} und die Dokumente D_1 bis D_n werden die entsprechenden Teilnetze mit $V_{\mathrm{DB},1}$ bis $V_{\mathrm{DB},n}$ bezeichnet. Diese Teilnetze mit m_{IN} Eingängen und m_{ID} Ausgängen erhalten als Eingabe das Suchwort und erzeugen als Ausgabe die ID des Dokumentes, falls das Suchwort im Dokument vorkommt. Kommt das Wort nicht im Dokument vor, erzeugen die Ausgänge 0-Bits.

Zusätzlich existieren Netze C_{merge}, die diese Teilergebnisse zusammenfassen. Diese erhalten als Eingaben die Ausgabewörter von je zwei Dokumententeilnetzen und geben als Ausgabe das erste Eingabewort aus, welches nicht nur aus 0-Bits besteht, sonst geben sie 0-Bits aus. Die Ergebnisse der Dokumententeilnetze werden durch diese Netze zusammengeführt, sodass die m_{ID} Ausgaben des letzten C_{merge}-Netzes die Ausgabe des gesamten Schaltnetzes V_{DB} darstellen.

Ein beispielhaftes Schaltnetz einer Datenbank mit vier Dokumenten sieht in verkürzter Formelnotation wie folgt aus:

$$V_{\mathrm{DB}} = C_{\mathrm{merge}}(V_{\mathrm{DB},4}, C_{\mathrm{merge}}(V_{\mathrm{DB},3}, C_{\mathrm{merge}}(V_{\mathrm{DB},2}, V_{\mathrm{DB},1})))$$

Auffällig ist hierbei die rekursive Schachtelung der C_{merge}-Netze.

Definition der Teilnetze

Teilnetze der Dokumente

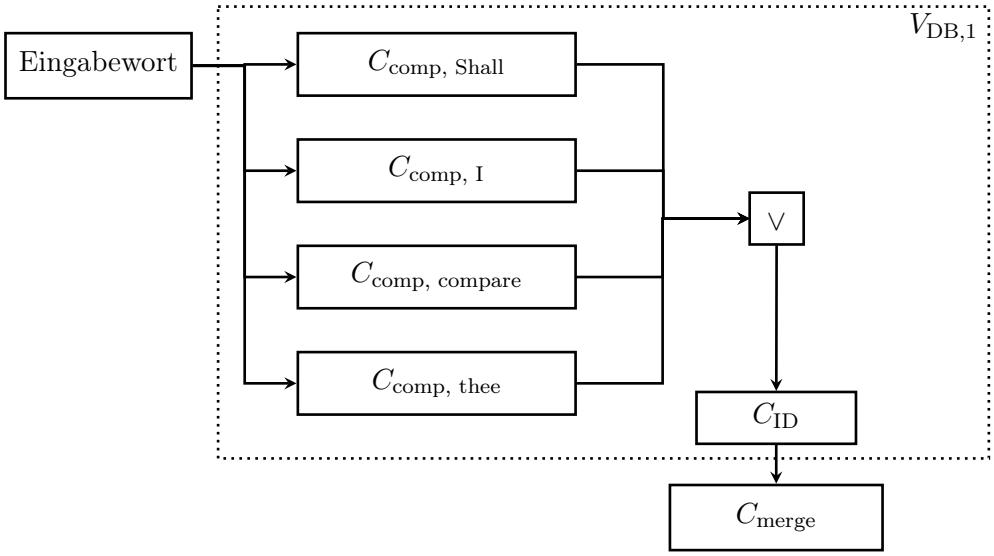

Abbildung 2: Übersicht eines Dokumententeilnetzes

Der Aufbau eines Dokumententeilnetzes ist dem des gesamten Netzes sehr ähnlich. Es setzt sich aus weiteren Teilnetzen zusammen, die jeweils für ein Wort des Dokumentes stehen. Sie erhalten das Suchwort als Eingabe und haben nur einen Ausgang, der genau dann ein 1-Bit erzeugt, wenn das Suchwort gleich dem eigenen Wort ist.

Konkret funktioniert das, indem die Bits des betrachteten Wortes jeweils mit den Bits des Suchwortes XOR-verknüpft werden und die Ergebnisse dieser Operationen Nor-verknüpft werden. Das Ergebnis der Nor-Verknüpfung ist das Ergebnis des Vergleichs. Für ein Wort mit der Binärrepräsentation »1001001« und die Bits b_1 bis b_7 des Suchwortes sieht ein Vergleichsnetz in Formelnotation wie folgt aus:

$$C_{\mathrm{comp,I}} = \overline{(1 \oplus b_1) \vee (0 \oplus b_2) \vee (0 \oplus b_3) \vee (1 \oplus b_4) \vee (0 \oplus b_5) \vee (0 \oplus b_6) \vee (1 \oplus b_7)}$$

Ein C_{comp}-Netz liefert also genau dann ein 1-Bit, wenn sein Wort dem Suchwort entspricht. Die Ausgaben der C_{comp}-Netze werden Oder-verknüpft und einem Netz C_{ID} zugeführt, das bei Eingabe eines 1-Bit die ID des Dokumentes ausgibt, sonst 0-Bits. Dieses Schaltnetz ist recht einfach und entsteht durch Und-Verknüpfung des Eingabebits mit der ID.

Es sei ein Dokument D_1 mit dem Volltext »Shall I compare thee« gegeben. Das Teilnetz für das Dokument sieht dann so aus:

$$V_{\mathrm{DB},1} = C_{\mathrm{ID}}(C_{\mathrm{comp,Shall}} \vee C_{\mathrm{comp,I}} \vee C_{\mathrm{comp,compare}} \vee C_{\mathrm{comp,thee}})$$

Teilnetze zum Zusammenführen

Die verwendeten Teilnetze C_{merge}, die Ausgaben von jeweils zwei Dokumentennetzen zusammenführen, unterscheiden sich untereinander nicht. Ein C_{merge}-Netz erhält die bitweise Kodierung von zwei potentiellen Dokumenten-IDs als Eingabe. Ist die Oder-Verknüpfung der Bits der ersten potentiellen ID kein 0-Bit (das Dokument ist also ein Treffer), wird diese ID an den Ausgängen erzeugt. Ansonsten wird die zweite potentielle ID an den Ausgängen erzeugt.

Wenn also mindestens ein Dokument das Suchwort enthält, wird durch das Zusammenführen die ID eines solchen Dokumentes am letzten C_{merge}-Teilnetz ausgegeben.

9 Diskussion des Suchkonzeptes

Durch die wiederkehrenden Bausteine und den rekursiven Aufbau ist es sehr einfach, aus einer Datenbank mit Textdokumenten eine Volltextsuche zu generieren oder zu erweitern. Für jedes hinzugefügte Dokument D_n wird ein weiteres Teilnetz $V_{\mathrm{DB,n}}$ sowie ein Teilnetz C_{merge} eingehängt. Die Größe des Schaltnetzes in Knoten steigt dabei mit der Anzahl der Dokumente n auf $\Theta(n)$ – angenommen die Anzahl der Wörter pro Dokument ist annähernd konstant. Analog dazu verhält sich auch die Tiefe des Netzes; durch das Einhängen von Dokumenten verlängert sich der längste einfache Pfad um die (konstante) Tiefe eines C_{merge}-Netzes.

Enthalten mehrere Dokumente das Suchwort, geht die Information über die zusätzlichen Treffer verloren. Um die IDs von mehreren Dokumenten auszugeben, müssten die C_{merge}-Netze stark erweitert werden. Außerdem ist die Anzahl der ausgegebenen potenziellen IDs durch die Volltextsuche fest vorgegeben, sodass die Möglichkeit, mehrere Ergebnisse auszugeben, ständige Gatterkosten verursacht. Eine einfachere – doch weniger nützliche – Möglichkeit ist es, die C_{merge}-Netze um einen zusätzlichen Ausgang zu erweitern, der genau dann ein 1-Bit erzeugt, wenn mehrfache Ergebnisse auftreten.

Es folgen weitere Vorschläge zur Erweiterung der Implementationsskizze:

Mehrwortsuchen

Es ist möglich, die gezeigte Volltextsuche um Mehrwortsuchen zu erweitern. Wir nehmen an, dass eine Suche nach n Wörtern durchgeführt werden soll. Dann erzeugen wir die n-fache Anzahl an Dokumententeilnetzen, wobei jedem Netz ein anderes Suchwort zugeführt wird. Die Ausgaben der Teilnetze werden Und-verknüpft und statt dem ursprünglichen Dokumententeilnetz dem C_{merge}-Netz zugeführt.

Statt der finalen Und-Verknüpfung kann auch ein logisches Oder verwendet werden. Dann muss nur noch ein Suchwort im Dokument enthalten sein, um einen Treffer auszulösen.

Verwendung Boolescher Algebra

Der Umbau auf Mehrwortsuchen kann weiterverwendet werden, um Boolesche Formeln in die Volltextsuche einzubauen. Statt die Dokumententeilnetze durch reine Und- bzw. Oder-Verknüpfungen zusammenzuführen, können komplexere Kombinationen eingesetzt werden.

Unscharfe Suchen

Um kleine Abweichungen der Suchwörter zuzulassen, können die C_{comp}-Netze abgewandelt werden. Statt der Oder-Verknüpfung, könnten die Abweichungen aufaddiert werden und mit einem Schwellwert verglichen werden.

Stemming – also die Reduzierung auf grammatikalische Grundformen – kann unterstützt werden, indem diese Reduzierung schon bei der Indexierung durchgeführt wird. Von den reduzierten Wörtern der Dokumente werden neue Formen abgeleitet, die dem Dokument hinzugefügt werden. Bei einer Suche sorgen diese abgeleiteten Formen für zusätzliche Treffer.

10 Ausblick

Das hier vorgestellte Schaltnetz bietet viel Raum für Erweiterungen und insbesondere Optimierungen. Ein invertierter Index könnte die Größe des Netzes reduzieren. Auch könnten $\mathcal{C}$-Ausführungsschemas verwendet werden, die auf Volltextsuchen optimiert sind.

Noch sind kryptografische Volltextsuchen nicht praxisrelevant, doch mit weiterer Forschung und Spezialhardware könnten sie es werden.

Literatur

[1] Frederik Armknecht, Colin Boyd, Christopher Carr, Kristian Gjøsteen, Angela Jäschke, Christian A. Reuter, and Martin Strand. A guide to fully homomorphic encryption. *IACR Cryptology ePrint Archive*, 2015:1192, 2015.

[2] Foteini Baldimtsi and Olga Ohrimenko. Sorting and searching behind the curtain: Private outsourced sort and frequency-based ranking of search results over encrypted data. *IACR Cryptology ePrint Archive*, 2014:1017, 2014.

[3] Lucas Barthelemy. A brief survey of fully homomorphic encryption, computing on encrypted data.

[4] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) lwe. In *In FOCS*, 2011.

[5] Gizem S. Çetin, Wei Dai, Yarkin Doröz, and Berk Sunar. Homomorphic autocomplete. *IACR Cryptology ePrint Archive*, 2015:1194, 2015.

[6] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 169–178, 2009.

[7] Hektor Haarkötter. *Google und mehr: Online-Recherche - Wie Sie exakte Treffer auf Ihre Suchanfragen erhalten.* Herbert von Halem Verlag, Köln, 2016.

[8] M. Joye and M. Tunstall. *Fault Analysis in Cryptography.* Information Security and Cryptography. Springer Berlin Heidelberg, 2012.

[9] Stefan Karzauninkat. *Die Suchfibel.* Ernst Klett Schulbuchverlag Leipzig GmbH, first edition, 1998.

[10] Yehuda Lindell. *Tutorials on the Foundations of Cryptography: Dedicated to Oded Goldreich.* Springer Publishing Company, Incorporated, 1st edition, 2017.

[11] Marlis Prinzing and Vinzenz Wyss. *Journalismus Atelier - Die richtige Recherche im Netz.* Europa Verlag AG Zürich, 2012.

[12] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *Proceedings of the Thirty-seventh Annual ACM Symposium on Theory of Computing*, STOC '05, pages 84–93, New York, NY, USA, 2005. ACM.

[13] R[onald] L. Rivest, A[di] Shamir, and L[eonard M.] Adleman. A method for obtaining digital signatures and public-key cryptosystems. *CACM*, 21(2):120–126. (This is the "RSA paper".).

[14] Tony Russell-Rose and Taylor Tate. *Designing the search experience - the information architecture of discovery.* Morgan Kaufmann, 2012.

[15] Heribert Vollmer. *Introduction to Circuit Complexity - A Uniform Approach.* Texts in Theoretical Computer Science. An EATCS Series. Springer, 1999.