TECHNISCHE
UNIVERSITÄT
DARMSTADT

Bachelor Thesis

Demonstrator für privatsphäre-schützende Gesichtserkennung

Nils Schroth

15. November 2016

CRISP
Center for Research
in Security and Privacy

Technische Universität Darmstadt
Center for Research in Security and Privacy
Engineering Cryptographic Protocols

Betreuer: Dr. Thomas Schneider
M.Sc. Daniel Demmler

Erklärung zur Abschlussarbeit gemäß §22 Abs. 7 APB der TU Darmstadt

Hiermit versichere ich, Nils Schroth, die vorliegende Bachelor Thesis ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

In der abgegebenen Thesis stimmen die schriftliche und elektronische Fassung überein.

Darmstadt, 15. November 2016

_____

Nils Schroth

Zusammenfassung

Um die öffentliche Sicherheit zu verbessern, setzen staatliche und private Sicherheitsunternehmen zunehmend auf Videoüberwachung. Jedoch wird dabei meist die Privatsphäre der Bevölkerung nicht ausreichend beachtet. In dieser Bachelor-Thesis stellen wir einen neuen Demonstrator für privatsphäre-schützende Gesichtserkennung vor. Der Demonstrator führt einen pirvatsphäre-schützenden Vergleich mit einem Bild eines Clients und Bildern einer Datenbank durch, die auf einem Server gespeichert sind. Anschließend gibt der Demonstrator aus, ob eine Übereinstimmung vorliegt, aber offenbart ansonsten keine weiteren Informationen an irgendeine Partei. Dieses Szenario hat mehrere Anwendungen, wie beispielsweise eine kamerabasierte Überwachung von öffentlichen Orten oder eine Identifikation an Grenzkontrollen. Heutzutage ist eine kamerabasierte Überwachung von öffentlichen Plätzen eine weit verbreitete Einsatzmöglichkeit, welche zu einer höheren Sicherheit führen soll, aber dabei gleichzeitig die Privatsphäre der Öffentlichkeit einschränkt. Daher stellen wir ein System vor, das sichere Zwei-Parteien-Berechnung nutzt, um zusätzlich die Privatsphäre der Öffentlichkeit zu schützen. Gleichzeitig muss die Gesichtserkennung schneller sein, als bei bisherigen Systemen, um sie auf reale Szenarien anwendbar zu machen.
Für diese Aufgabe kann SCiFI (SSP'10), ein System zur sicheren Berechnung der Gesichtserkennung verwendet werden. Aber der Abgleich der Bilder, welche durch Vektoren repräsentiert werden, wird von SCiFI ursprünglich durch die Berechnung der Hamming-Distanz auf Basis von homomorpher Verschlüsselung durchgeführt, was eine hohe Laufzeit und Kommunikationskomplexität aufweist. Wir ersetzten diesen Teil durch eine privatsphäre-schützende Hamming-Distanz-Berechnung mit dem ABY-Framework (NDSS'15), was zu einer Leistungsverbesserung führt. ABY ist ein mixed-protocol Framework für die sichere Zwei-Parteien-Berechnung, die es erlaubt, sichere Berechnungsschemata zu kombinieren und effizient zwischen ihnen zu konvertieren. Die Hamming-Distanz-Berechnung basiert auf einem Boolean Circuit, der durch Yao's Garbled Circuit-Protokoll oder dem GMW-Protokoll evaluiert werden kann. Die Merkmalsextraktion von Gesichtern basiert weiterhin auf SCiFI, da sie für eine sichere Berechnung entworfen wurde und eine hohe Erkennungsrate aufweist. Die Auswertung der Circuits für die Hamming Distanz ist sehr schnell, so dass der neue Demonstrator eine deutliche Verbesserung der Laufzeit im Vergleich zu dem ursprünglichen SCiFI-System ist.

Abstract

In order to improve public safety, government and private security companies increasingly put on video surveillance, which threatens the privacy of the public. In this Bachelor-Thesis we introduce a new demonstrator for privacy-preserving face recognition. The demonstrator performs privacy-preserving face recognition on a client input image with a database of images which outputs whether there is a match but reveals no other information to any party. This scenario has multiple applications, such as a camera-based surveillance of public places or identification at border control. Nowadays a camera-based surveillance of public places is a security feature, which can lead to a higher security but also reduces the privacy of the public. Therefore, we propose a system that uses secure two-party computation which additionally protects privacy of the public. Simultaneously, the face recognition needs to be faster to make them applicable to real-world scenarios.
For this task SCiFI (SSP'10), a system for Secure Computation of Face Identification can be used. But the recognition part of SCiFI is originally done by computing the hamming distance based on homomorphic encryption on vectors, each representing an image, what has a high runtime and communication complexity. We replaced this part by a secure hamming distance computation using the ABY-Framework (NDSS'15), which results in a performance improvement. ABY is a mixed-protocol framework for secure two-party communication, which allows to combine secure computation schemes and efficiently convert between them. The hamming distance computation is based on boolean circuit and can be evaluated with Yao's garbled circuit-Protocol or with the GMW-Protocol. The feature extraction of faces is still based on SCiFI, since it is well designed for secure computation and has a high recognition performance. The evaluation of the circuit for the hamming distance is very fast, so the new Demonstrator is a significant improvement in runtime compared to the original SCiFI system.

Inhaltsverzeichnis

1 Einleitung

Seit dem Beginn des 21. Jahrhunderts haben immer wieder verschiedene Terroranschläge, wie beispielsweise die Anschläge auf das World Trade Center vom 11. September 2001 in New York oder die Anschläge in Paris 2015, die Welt erschüttert. Dadurch wurden heftige Diskussionen ausgelöst, wie die Sicherheit der Öffentlichkeit erhöht werden kann und in wie weit Attentäter bereits vor dem Verüben eines Anschlags gestoppt werden können. Gleichzeitig muss dabei aber auch die Privatsphäre der Gesellschaft geschützt werden. Diese beiden Aspekte stehen in Konflikt zueinander. Es gilt daher eine optimale Lösung zu finden, die beide Positionen gleichermaßen berücksichtigt. Dabei gilt es, die Freiheit und Privatsphäre des Einzelnen zu bewahren und dem Staat die nötigen Rechte und Mittel einzuräumen, die Sicherheit der Allgemeinheit zu gewährleisten.

Um die Sicherheit zu erhöhen und Terroranschläge möglichst im Vorfeld zu verhindern, gibt es verschiedene technische Lösungsansätze, die der Staat oder private Sicherheitsfirmen einsetzen können. Dazu zählen unter anderem die Überwachung der Telekommunikation, des Internet-Verkehrs und die Video-Überwachung öffentlicher Orte. Bei all diesen Einsatzmöglichkeiten wird jedoch nicht nur in die Privatsphäre der potenziell Verdächtigen, sondern immer auch in die von unbeteiligten Personen eingegriffen. In dieser Arbeit liegt der Fokus auf der privatsphäre-schützenden Gesichtserkennung basierend auf sicherer Zwei-Parteien Berechnung (S2C), welche bei der Videoüberwachung angewendet werden kann. Andere Lösungsansätze zur Erhöhung der öffentlichen Sicherheit werden hier nicht weiter betrachtet.

1.1 Motivation

Die Gesichtserkennung durch Videoüberwachung kann sowohl zur Prävention, als auch zur Intervention genutzt werden. Der präventive Aspekt der Videoüberwachung liegt in der Abschreckung von möglichen Gewalttätern und Terroristen. Durch die Videoüberwachung besteht die Möglichkeit der Identifizierung und somit der Strafverfolgung, wodurch das Sicherheitsbefinden der Bevölkerung verstärkt und gleichzeitig eine abschreckende Wirkung erzeugt werden kann.
Andererseits kann die Videoüberwachung auch interventiv zum Einsatz kommen. Dies ist zum einen möglich, um einzugreifen, wenn man verdächtige Vorgänge beobachtet, wie zum Beispiel, wenn Personen an öffentlichen Plätzen ein Gepäckstück zurück lassen. Zum anderen, um mittels Gesichtserkennung Personen zu identifizieren, die bereits in einer Datenbank für

Verdächtige erfasst worden sind.
Eine ständige Echtzeit-Beobachtung erfordert einen immensen Personaleinsatz und somit auch einen hohen Kostenaufwand. Allerdings benötigt diese Form der Überwachung einen geringen Einsatz von spezieller Technik, da normale Überwachungskameras ausreichend sind. Aus diesem Grund ist diese Form des interventiven Eingriffs nicht effizient umsetzbar und daher praktisch nicht flächendeckend einsatzfähig. Anders sieht dies hingegen bei der computergestützten Gesichtserkennung aus. Hierbei ist der Personalaufwand sehr gering, wodurch auch der finanzielle Aufwand auf Dauer niedrig gehalten werden kann. In Zukunft wird der Überwachungsaufwand und die Überwachungsdichte immer weiter ansteigen, um eine großflächige Sicherheitsüberwachung zu gewährleisten, die nahezu alle öffentlichen Bereiche abdeckt. Allerdings darf dabei die Privatsphäre der Bevölkerung nicht außer Acht gelassen werden.
Zudem spielt in diesem Zusammenhang auch der Aspekt der Datenspeicherung eine Rolle. Durch aktuell eingesetzte Systeme für Gesichtserkennung durch Videoüberwachung ist es möglich, zu jeder Person eine Historie aufzustellen und ihre Positionen zu verfolgen. Diese „Big Brother“-Überwachung wird in der Gesellschaft weitestgehend abgelehnt, da Personen unabhängig einer Verdächtigung und teilweise gegen ihren Willen überwacht, überprüft und personenbezogene Daten gespeichert werden. Daher gilt es eine Form der Gesichtserkennung zu entwickeln, welche keine personenbezogenen Daten erhebt und speichert, sondern lediglich anonym die aufgenommen Bilder mit Listen von Verdächtigen in Echtzeit abgleicht. Dadurch wird die Privatsphäre der Bevölkerung in größtmöglicher Weise geschützt und dennoch ein hohes Maß an Sicherheit gewährleistet.

Um diese Aufgabe zu bewältigen, ist es nötig das Fachwissen des Gebietes der Gesichtserkennung mit privatsphäre-schützender Berechnung zu vereinen, wie zum Beispiel der S2C. Gesichtserkennung kann mittels verschiedener Algorithmen durchgeführt werden, wie Eigenfaces [TP91] oder dem extra für S2C entwickelten SCiFI [OPJM10]. S2C ermöglicht es, den Abgleich von öffentlich Aufnahmen mit einer Liste von Verdächtigen durchzuführen, ohne die Privatsphäre der Öffentlichkeit zu beeinträchtigen oder die Vertraulichkeit der Liste der Verdächtigen zu gefährden. Dafür wird ein Client-Server-Model verwendet, wobei die Kamera der Client ist und die Liste mit Verdächtigen auf dem Server liegt. Bei einem Abgleich wird lediglich das Ergebnis, ob ein Treffer vorliegt, an eine der beiden Parteien oder auch an beide Parteien übertragen, je nach Anwendungsfall. Außer diesem Ergebnis erhalten die Parteien keine weiteren Informationen von der jeweils anderen Partei.
Für den sicheren Abgleich können verschiedene S2C-Protokolle in Betracht gezogen werden, wie homomorphe Verschlüsselung (HE) [RAD78], das Goldreich-Micali-Widgerson-Protokoll (GMW) [GMW87] oder Yao's Garbled Circuits (GC) [Yao86]. Ein möglicher Anwendungsfall für privatsphäre-schützende Gesichtserkennung ist die automatische Identifizierung von Gefährdern bei Grenzkontrollen. In diesem Fall wäre eine Übermittlung des Ergebnis an den Client oder eine dritte Partei sinnvoll, um bei einem Treffer mit der Datenbank direkt eingreifen zu können. Bei der Überwachung von mehreren öffentlichen Plätzen ist hingegen eine Übermittlung des Ergebnis an den Server sinnvoll, um eine sinnvolle Koordinierung von Einsätzen zu ermöglichen. SCiFI ist ein System, welches für solche Szenarien genutzt werden kann. Jedoch ist es zu langsam für den Echtzeit Betrieb, da die S2C nicht state-of-the-art ist

und einige Features fehlen, wie die Verwaltung der Verdächtigen-Liste oder die Auswahl, wer das Ergebnis erhalten soll.

1.2 Beitrag

In dieser Arbeit wird ein Demonstrator vorgestellt, der state-of-the-art Techniken der Gesichtserkennung und der sicheren Zwei-Parteien Berechnung nutzt. Aufgrund der schlechten Performanz des ursprünglichen SCiFI Systems ist dieses in der Praxis nicht sinnvoll anwendbar.

Da der in SCiFI entwickelte Merkmalsextraktions-Algorithmus jedoch im Vergleich zu anderen Algorithmen am besten für privatsphäre-schützende Gesichtserkennung geeignet ist, wird die SCiFI Implementierung in dieser Arbeit als Grundlage verwendet. Der Vorteil dieser Methode liegt in der Repräsentation der Merkmalsvektoren und der Abstandsfunktion. Es werden diskrete Werte mit fester Größe unabhängig von der Bildgröße verwendet. Zur Berechnung der Ähnlichkeit zweier Bilder wird die Hamming-Distanz (HD) ermittelt, welche effizienter zu berechnen ist, als die Euklidische Distanz (ED). Um das System praktisch anwendbar zu machen, musste der S2C-Teil ausgetauscht werden, da die vorliegende Implementierung von HE in Java nicht mehr mit aktuellen Techniken zur S2C mithalten kann. Daher wird im vorliegenden Fall für diesen Teil das ABY-Framework [DSZ15] eingesetzt. Das Framework wird genutzt, um die privatsphäre-schützende HD-Berechnung durchzuführen, die für die Distanzmessung von Merkmalen in SCiFI benötigt wird. Dafür wurde ein neuer Boolean Circuit (BC) implementiert, der die HD berechnet, anschließend das Minimum bestimmt und dann einen Vergleich mit einem Schwellwert durchführt. Da SCiFI in Java und ABY in C/C++ implementiert ist, wurde eine Interprocess Communication eingerichtet, sodass beide Teile über STD I/O kommunizieren können. Der neue Demonstrator nutzt so die Vorteile des SCiFI-Systems und ist gleichzeitig schneller, als es vorhandene privatsphäre-schützende Gesichtserkennungs-Demonstratoren bisher waren. Außerdem bietet der Demonstrator ein verbessertes und benutzerfreundliches DB-Managment Tool, welches den praktischen Einsatz erleichtert. Des Weiteren kann ausgewählt werden, welche Partei das Ergebnis erhalten soll und es kann clientseitig der Schwellwert für den Abgleich variabel gesetzt werden.

1.3 Gliederung

In Kapitel 2 werden die Notationen und der theoretische Hintergrund der Protokolle sowie die Gesichtserkennungs-Algorithmen beschrieben. Danach folgt ein Überblick über ähnliche Arbeiten im Bereich der Gesichtserkennungs-Algorithmen in Kapitel 3. Das ABY Framework und der für SCiFI notwendige Minimum Hamming Distance Circuit werden im Kapitel 4 vorgestellt. Zum Schluss folgt in Kapitel 5 die Dokumentation des neuen auf SCiFI und ABY basierenden Demonstrators und ein Vergleich mit dem Original SCiFI.

2 Grundlagen und Rahmenbedingungen

In diesem Kapitel werden theoretische Grundlagen beschrieben, die im weiteren Verlauf der Arbeit relevant sind. Dazu zählt das Sicherheitsmodell, die Abstandsfunktionen, die für den Abgleich von Gesichtern verwendet werden und die S2C-Grundlagen. Zwei Fundamente für S2C sind Boolean Circuits (BC) und Oblivious Transfer (OT) sowie Multiplication Triples (MT). Es werden zwei darauf aufbauende Ansätze für S2C dargestellt, das GMW-Protokoll und Yao's GC-Protokoll. Außerdem wird die homomorphe Verschlüsselung (HE) kurz eingeführt. Abschließend werden noch die Gesichtserkennungs-Algorithmen vorgestellt, welche die Grundlage für viele Arbeiten in diesem Bereich bilden.

2.1 Sicherheitsmodell

Durch das Sicherheitsmodell werden Angriffe von Teilnehmern eines Protokolls klassifiziert. Es gibt den aktiven (malicious) und den passiven (semi-honest) Angreifer. Durch die Angabe einer der beiden Angreifer, wird die Sicherheit eines Protokolls beschrieben. Dabei versucht der Angreifer im Verlauf des Protokolls unautorisiert Informationen zu bekommen.

Der passive Angreifer weicht nicht vom Ablauf des Protokolls ab, versucht jedoch aus den empfangenen Nachrichten zusätzliche Informationen zu bekommen, die das Protokoll eigentlich verbergen will. Dieses abgeschwächte Modell erlaubt es effiziente Protokolle zu entwerfen und ein Mindestmaß an Sicherheit zu gewährleisten.
Der aktive Angreifer kann beliebig vom Ablauf des Protokolls abweichen. Im Vergleich zum passiven Angreifer kann er beliebige Nachrichten schicken und das Protokoll unterbrechen. Dadurch ist es schwieriger ein Protokoll sicher gegen dieses Angriffsmodell zu entwerfen. Daher sind die meisten Protokolle nur gegen passive Angriffe sicher, da sie wesentlich effizienter in Laufzeit und Kommunikation entworfen werden können. Die in dieser Arbeit vorgestellten Protokolle sind sicher gegenüber passiven Angreifern. Es existieren jedoch Erweiterungen für aktive Sicherheit, die mit den hier gezeigten Techniken kompatibel sind.

2.2 Abstandsfunktionen

An dieser Stelle werden die im weiteren Verlauf benutzten Abstandsfunktionen definiert und erklärt. Die Euklidische Distanz und die Hamming-Distanz dienen als Maß für den Abgleich

von Merkmalsvektoren bei Eigenfaces beziehungsweise SCiFI. Eine kleinere Distanz bedeutet eine höhere Ähnlichkeit der Bilder, die den Vektoren entsprechen. Das Skalarprodukt wird für die Generierung eines neuen Merkmalsvektors bei Eigenfaces benötigt.

2.2.1 Skalarprodukt

Das Skalaprodukt (SP) oder auch inneres Produkt von zwei K-dimensionalen Vektoren $X = (X_1, ..., X_K)$ und $Y = (Y_1, ..., Y_K)$ ist definiert als

$$\mathrm{SP}(X, Y) = \sum_{i=1}^{K} X_i Y_i.$$

Es verknüpft zwei K-dimensionale Vektoren in einem K-dimensionalen Vektorraum zu einem Zahlenwert (Skalar). Dabei hängt das Ergebnis von der Länge der beiden Vektoren und dem von ihnen eingeschlossenen Winkel ab.

2.2.2 Euklidische Distanz

Die quadratische euklidische Distanz (ED) von zwei K-dimensionalen Vektoren $X = (X_1, ..., X_K)$ und $Y = (Y_1, ..., Y_K)$ ist definiert als

$$\mathrm{ED}(X, Y) = \sum_{i=1}^{K} (X_i - Y_i)^2 = \sum_{i=1}^{K} ((X_i)^2 - 2X_i Y_i + (Y_i)^2).$$

Die Distanz gibt die Länge der Verbindung von zwei K-dimensionalen Vektoren in einem K-dimensionalen Vektorraum an. Wenn zwei Vektoren im Raum nahe beieinander liegen, ist der Distanzwert dementsprechend klein.

2.2.3 Hamming-Distanz

Die Hamming Distanz (HD) von zwei ℓ-bit Vektoren $X = (x_1, ..., x_\ell)$ und $Y = (y_1, ..., y_\ell)$ ist definiert als

$$\mathrm{HD}(X, Y) = \sum_{i=1}^{\ell} x_i \oplus y_i.$$

Sie berechnet die Anzahl an unterschiedlichen Bitwerten in zwei ℓ-bit Vektoren. Dafür wird die Summe der Ergebnisse der bitweisen XOR-Operation genutzt. Die Distanz von zwei Vektoren ist niedrig, wenn sie sich nur an wenigen Stellen unterscheiden. Im Vergleich zu den beiden vorherigen Metriken arbeitet die HD nur auf diskreten Werten, welche effizienter, ohne Verlust der Genauigkeit, privatsphäre-schützend zu berechnen sind.

2.3 Sichere Zwei-Parteien Berechnung

Die hier vorgestellten S2C-Protokolle basieren auf Oblivious Transfer, Boolean Circuits und Multiplication Triples. Diese Grundlagen werden kurz erklärt und anschließend werden zwei Techniken zur S2C vorgestellt, die in dieser Arbeit verwendet werden. Das sind GC und GMW.

2.3.1 Oblivious Transfer

Oblivious Transfer ermöglicht den Austausch von Informationen zwischen einem Sender und einem Empfänger. Dabei hält der Sender N Zeichenketten $X_0, \ldots, X_{N-1}$ mit Länge ℓ als Eingabewerte und der Empfänger einen m-bit Auswahlvektor b der die Indizes der Nachrichten enthält, die er empfangen möchte. Dabei muss $m < N$ gelten. Der Empfänger kann durch den Auswahlvektor also eine oder mehrere Zeichenketten empfangen. Dabei gilt, dass der Sender keine Informationen darüber hat, welche Zeichenketten der Empfänger empfangen hat und gleichzeitig der Empfänger keine Informationen über die nicht ausgewählten Zeichenketten des Senders erhält.

Es gibt zwei Notationen, um OT-Protokolle zu beschreiben. Bei einer Form erfolgt die Benennung nach der Anzahl der Zeichenketten, die der Sender hält und der Länge des Auswahlvektors b. Bei der 1-out-of-2 OT, geschrieben als OT $\binom{2}{1}$, hält der Sender zwei Zeichenketten und der Empfänger wählt eine davon aus. Bei der zweiten Form wird die Länge der Zeichenketten ℓ und die Anzahl der Aufrufe m des OT angegeben und die Anzahl der Zeichenketten des Senders und die davon empfangen Nachrichten für den Empfänger vorangestellt. Bei m Aufrufen auf ℓ-bit Zeichenketten einer 1-out-of-2 OT, wird es als $\binom{2}{1}$-OT$_\ell^m$ geschrieben. Im Folgenden benutzen wir die erste Notation.

Abbildung 2.1 zeigt eine schematische Darstellung des Protokolls. Für Yao's GC-Protokoll wird ein OT $\binom{2}{1}$-Protokoll zum Austausch der verschlüsselten Schlüssel verwendet, wohingegen bei dem GMW-Protokoll ein OT $\binom{4}{1}$-Protokoll oder zwei Aufrufe eines OT $\binom{2}{1}$-Protokoll separat für jedes Bit zur Evaluation der AND-Gatter benötigt werden.

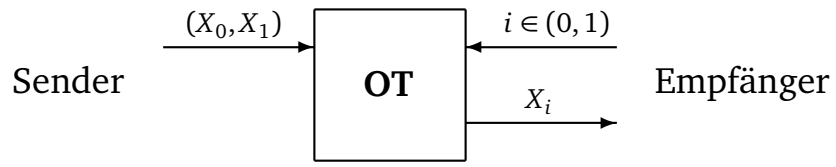

Abbildung 2.1: 1-out-of-2 Oblivious Transfer

OT-Protokolle sind seid ihrer Einführung durch Rabin 1981 [Rab81] ein viel diskutiertes Forschungsthema. Daher gibt es einige Erweiterungen und Verbesserungen, wie das OT Extension Protokoll [IKNP03], welche die meisten teuren Public-Key Operationen durch effizientere symmetrische kryptographische Funktionen ersetzen, so dass nur noch wenige Public-Key Operationen nötig sind. Dadurch wird die benötigte Rechenzeit stark reduziert, sodass die Netzwerk-Bandbreite zum Flaschenhals wird. Eine weitere Erweiterung ist random OT (R-OT) [ALSZ13]. Bei dieser Variante werden die Eingaben vom Sender gleichmäßig und zufällig während des OT gewählt und die Nachricht vom Sender zum Empfänger fällt weg. Außerdem gibt es noch correlated OT (C-OT) [ALSZ13] von Schneider et al., bei welchem die Kommunikations-Komplexität um den Faktor zwei optimiert wurde. Dabei hält der Sender eine Korrelation Δ als Eingabe und erhält als Ausgabe einen zufälligen Wert und einen zweiten Wert, der durch die Korrelation Δ bestimmt wird.

2.3.2 Multiplication Triples

Multiplication Triples [Bea91] wurden 1992 von Beaver eingeführt und sind eine alternative Form zur Auswertung von AND-Gattern, wie sie im GMW-Protokoll eingesetzt werden. Ein MT besteht aus drei zufälligen Shares a_i, b_i, c_i mit $i \in \{1, 2\}$ welche die Gleichung

$$c_1 \oplus c_2 = (a_1 \oplus a_2) \wedge (b_1 \oplus b_2)$$

erfüllen. Da sie unabhängig von der Eingabe der beiden Parteien P_1 und P_2 sind, können sie bereits in der Setup-Phase des Protokolls generiert werden. Dafür kann ein OT $\binom{4}{1}$ verwendet werden, bei dem P_1 nur die Shares a_1, b_1, c_1 und P_2 nur a_2, b_2, c_2 erhält.
Diese MTs werden während der Online-Phase des Protokolls zur Maskierung der Eingabe-Shares der AND-Gatter verwendet. Die Parteien mit Eingaben x_i, y_i mit $i \in \{1, 2\}$ berechnen $d_i = a_i \oplus x_i$ und $e_i = b_i \oplus y_i$, tauschen d_i, e_i aus und leiten daraus $d = d_1 \oplus d_2$ und $e = e_1 \oplus e_2$ her. Damit werden dann die Ausgabe-Shares des AND-Gatters $w_1 = (d \wedge e) \oplus (d \wedge b_1) \oplus (e \wedge a_1) \oplus c_1$ und $w_2 = (d \wedge b_2) \oplus (e \wedge a_2) \oplus c_2$ berechnet.
Es muss für jedes AND-Gatter ein neues MT zur Maskierung verwendet werden, jedoch führt das zu einer deutlich schnelleren Online-Phase, da hier jede Partei nur noch ein Share pro AND-Gatter versenden muss, statt ein OT $\binom{4}{1}$ auszuführen.

2.3.3 Boolean Circuits

Boolean Circuits dienen dazu, beliebige Funktionen durch logische Gatter und Leitungen zwischen diesen zu repräsentieren. Dabei wird die Funktion durch eine Aneinanderreihung von bitweisen Operationen berechnet. Solche Operationen werden durch Gatter dargestellt. Es gibt die binären AND-, OR-, XOR-Gatter und das unäre NOT-Gatter, aus welchen auch Kombinationen verfügbar sind, wie z.B. das NAND-Gatter.
Diese Gatter haben bei dem unären NOT-Gatter eine Ein- und Ausgabe und bei den übrigen

Gattern zwei Eingaben und eine Ausgabe, welche entsprechend einer Wahrheitstabelle spezifiziert sind. Die binären Werte für die Ein- und Ausgabe liegen auf Leitungen, die wiederum eine Ausgabe aus einem Gatter oder Eingabe für ein weiteres Gatter sein können. Außerdem gibt es Multiplexer Gatter, die aus mehreren Eingaben eine Ausgabe entsprechend einer Selektionseingabe weitergeben.
Ein solcher BC besteht aus m Eingaben, n Ausgaben und g Booleschen Gattern, wobei ein BC so gestaltet werden kann, dass er eine Funktion nur durch AND- und XOR-Gatter darstellt. Diese Repräsentation wird für die im Folgenden vorgestellten S2C-Protokolle verwendet. Abbildung 2.2 zeigt beispielhaft einen BC mit $m = 5, n = 2, g = 4$.

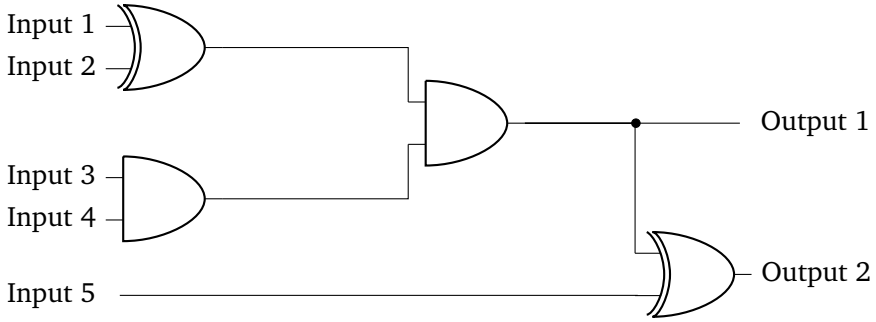

Abbildung 2.2: Boolean Circuit Beispiel

2.3.4 Yao's Garbled Circuits

Yao's Garbled Circuit Protokoll [Yao86] wurde 1982 von Andrew C. Yao eingeführt. Es ermöglicht zwei Parteien die sichere Evaluierung von beliebigen Funktionen, die als BC repräsentiert werden. Das Protokoll verwendet effiziente symmetrische Kryptographie und das oben angesprochene 1-out-of-2 OT Protokoll.

Bei diesem Protokoll gibt es einen Creator und einen Evaluator. Der Creator stellt die zu berechnende Funktion als BC dar, aus welchem er dann den GC erstellt. Dafür wird für jedes Gatter des BC eine Garbled Table aufgestellt, in der für die beiden Eingänge des Gatters zwei zufällige Schlüssel stehen, die den Werten 0 und 1 entsprechen. Die Einträge der Tabellen werden zufällig vertauscht, um keine Rückschlüsse auf die Werte zuzulassen, da es bei zwei Eingaben nur vier mögliche Kombinationen gibt. Diese Tabellen werden zusammen mit den Schlüsseln der Eingaben des Creators an den Evaluator gesendet. Der Evaluator benötigt jedoch auch noch die Schlüssel für seine eigenen Eingaben, um den Ausgang der Gatter zu berechnen. Da der Evaluator seine Eingaben ebenfalls geheim halten möchte, bekommt er seine Schlüssel durch das 1-out-of-2 OT-Protokoll, dass für jedes AND-Gatter ausgeführt werden muss. Dabei hält der Creator beide möglichen Schlüssel und der Evaluator wählt den richtigen Schlüssel aus, wodurch der Creator keine Information über die Eingabe des Evaluator erhält. So wird der GC schrittweise für jedes Gatter ausgewertet, wobei die Ausgänge eines ausgewerteten Gatters als Eingabe für ein anderes Gatter stehen können. Der Evaluator erhält

am Ende des Circuits eine verschlüsselte Ausgabe. Je nachdem, ob entweder der Creator, der Evaluator oder Beide die entschlüsselte Ausgabe erhalten sollen, muss der Creator sein Mapping zur Entschlüsselung an den Evaluator oder der Evaluator das verschlüsselte Ergebnis an den Creator senden.

Die Optimierung von Kolesnikov und Schneider aus dem Jahr 2011 [KS08] erlaubt es, XOR-Gatter nahezu ohne Rechenaufwand auszuwerten, da der Creator sie nicht mehr garblen muss. Außerdem gibt es noch garbled row reduction [NPS99; PSSW09], welche die Einträge der Garbled Table auf 3 statt 4 reduziert. In [HEKM11] wird eine Optimierung vorgestellt, die den Ablauf des Protokolls durch Pipelining effizienter macht. Dabei wird die Evaluierung des Circuits ausgeführt, sobald einzelne Gatter oder Teile des BC gegarbled sind und es muss nicht gewartet werden, bis der gesamte BC gegarbled ist. Dadurch muss auch nicht mehr der gesamte Garbled Circuit gespeichert werden.

2.3.5 GMW

Ein anderer Ansatz zur Evaluierung von Boolean Circuits ist das 1987 von Goldreich, Micali und Wigderson eingeführte GMW-Protokoll [GMW87]. Statt verschlüsselter Gatter werden in diesem Protokoll jedoch Shares zur Evaluierung genutzt.

Ein Share wird durch Secret Input Sharing erzeugt, bei dem beide Parteien S und C ihre Eingaben x und y mit zufälligen Bitstrings x_r und y_r durch die XOR-Operation $x_C = x \oplus x_r$ und $y_S = y \oplus y_r$ maskieren und die Ergebnisse x_C und y_S der jeweils anderen Partei mitteilen. Der Server kennt anschließend y_S und seinen eigenen zufälligen Bitstring x_r und der Client dementsprechend x_C und y_r. Diese Shares werden als Eingabe für den BC benutzt.

Das Protokoll kann in zwei Phasen aufgeteilt werden. Die Setup-Phase, welche lokale Vorberechnungen vornimmt, und die Online-Phase, in welcher die zwei Parteien kommunizieren, um die Funktion auszuwerten. Da die Setup-Phase die zeitkritische Phase ist, kann die eigentliche Auswertung in der Online-Phase sehr schnell durchgeführt werden.

Als erstes müssen beide Parteien ihre Eingaben durch XOR secret sharen. Dafür wählen beide Parteien zufällige Bitstrings und berechnen das XOR mit ihren Eingaben. Das Ergebnis teilen sie der jeweils anderen Partei mit. Da XOR assoziativ ist, können XOR-Gatter während des Protokolls einfach lokal auf dem eigenen Share und dem der anderen Partei ausgewertet werden. Für AND-Gatter wird ein 1-out-of-4 OT Protokoll benötigt, bei welchem eine Partei die möglichen Ausgaben vorberechnet und die andere Partei dann die Ausgabe entsprechend zu ihren Eingaben auswählt. Außerdem können AND-Gatter auch mit MTs evaluiert werden. Das finale Ergebnis der Funktion erhalten beide Parteien in dem sie ihre letzten Ergebnisse des BC miteinander teilen und mit den eigenen Ergebnissen das XOR berechnen.

In [CHK+12] wurde gezeigt, dass das OT $\binom{4}{1}$-Protokoll parallelisiert werden kann, was zu einem Performanz-Gewinn führt. Ein effizienterer Ansatz zum Berechnen der AND-Gatter sind MTs, siehe dazu Abschnitt 2.3.2.

2.4 Homomorphe Verschlüsselung

1978 wurde das Konzept der homomorphen Verschlüsselungen von Rivest, Adleman und Dertouz eingeführt [RAD78]. Dadurch ist es möglich, auf verschlüsselten Daten zu rechnen, ohne den Schlüssel zu kennen, wobei die Operationen in der entsprechenden Ciphertext-Struktur stattfinden. Zwei Beispiele für HE sind Paillier [Pai99] und Dåmgard-Geisler-Krøigaard (DGK) [DGK08], welche nur additiv homomorph sind. Das bedeutet, dass nur die Addition von verschlüsselten Werten und die Multiplikation von einem verschlüsselten Wert mit einer öffentlichen Konstanten durchgeführt werden kann. Volle HE, die alle Operationen beherrscht, ist noch nicht praktisch einsetzbar. Ein Vorteil von HE ist die Wiederverwendbarkeit der Ciphertexte, wodurch die Kommunikation zwischen den beiden Parteien reduziert wird. Ein Nachteil sind die public-key Operationen, auf denen HE aufbaut, welche für große Sicherheitsparameter nicht mehr effizient berechenbar sind.

2.5 Gesichtserkennungs-Algorithmen

Im Folgenden werden die beiden bekanntesten Gesichterkennungs-Algorithmen zur privatsphäre-schützenden Gesichtserkennung vorgestellt, Eigenfaces und SCiFI. Bei der Gesichtserkennung geht man davon aus, dass es einen Client mit Bild X und einen Server mit einer Datenbank an Bildern $Y_1, ..., Y_N$ gibt. Die Algorithmen versuchen zu gegebenem Bild X das ähnlichste Bild in der Datenbank Y zu finden und als Ergebnis auszugeben, falls die Ähnlichkeit, welche als Distanz ausgedrückt wird, über einem bestimmten Schwellwert liegt. Dafür verwenden die beiden Algorithmen unterschiedliche Distanz-Metriken. Um einen solchen Abgleich durchzuführen, müssen aber zunächst alle Bilder in einer entsprechenden Form vorliegen, um sie vergleichen zu können. Diese Repräsentation erhält man durch die algorithmusspezifische Merkmalsextraktion, welche aus einer Bild-Datei einen Vektor erstellt.

2.5.1 Eigenfaces

Matthew Turk und Alex Pentland stellten 1991 in [TP91] ihren Ansatz zur effizienten Gesichtserkennung vor, welcher als Eigenfaces bekannt ist. Bei diesem Algorithmus werden Bilder auf einen niedrig-dimensionalen Vektorraum abgebildet. Die Basen für den Vektorraum werden durch Principal Component Analysis (PCA) aus einem Trainingsset erstellt.
Die PCA ist eine lineare Projektion, die die ursprünglichen Bild-Vektoren in den angesprochenen Vektorraum abbildet. Dafür werden die Variablen durch eine kleinere Zahl an Hauptkomponenten dargestellt, wodurch zwar Informationen verloren gehen, aber nur solche, die den

geringsten Einfluss auf die Verteilung haben. Diese Hauptkomponenten sind Linearkombinationen der ursprünglichen Variablen. Dabei wird die Verteilung der Testbilder maximiert, um ein Bild leichter einem ähnlichen Bild zuweisen zu können, da es weniger Überschneidungen gibt.
Die daraus entstehenden Vektoren nennt man Merkmalsvektoren, die ein Bild für einen Abgleich optimiert darstellen. Für den Abgleich kann der Algorithmus in drei Schritte aufgeteilt werden. Als erstes findet eine Projektion des Client Bildes in den Vektorraum des Servers statt. Danach wird die euklidische Distanz zu allen Merkmalsvektoren berechnet und zum Schluss wird die geringste Distanz mit einem Schwellwert verglichen.

Merkmalsextraktion

Für den Algorithmus werden zunächst alle Bilder auf eine feste Größe geschnitten, bei welcher das Gesicht in der Mitte liegt. Die in [EFG+09] verwendete Bildgröße von 92 x 112 Pixeln wurde in darauffolgenden Arbeiten als Standard für die Berechnung von Eigenfaces angenommen. Daraus ergibt sich ein Vektor mit $n = 10304$ Dimensionen, der für jedes Pixel einen Grauwert enthält. Ausgehend von einem Trainingsset aus beliebigen Vektoren wird dann die Kovarianzmatrix C aufgestellt. Durch anwenden der PCA auf die Matrix C werden die Eigenwerte und Eigenvektoren des Trainingssets berechnet. Aus diesen Eigenvektoren werden die $K << M$ Eigenvektoren $u_1, ..., u_K$ mit den größten Eigenwerten ausgewählt, um den niedrig-dimensionalen Vektorraum aufzuspannen. Die Basisvektoren dieses Raums nennt man Eigenfaces. In diesen Raum werden die Bilder für den späteren Abgleich projiziert, welche jeweils durch 10304 Zeilen Vektoren repräsentiert werden.

Zur Projektion wird mit

$$\phi = X - \psi$$

die Abweichung des Bildvektors vom Durchschnittsvektor ψ, der aus dem Trainingsset bestimmt wurde, berechnet. Mit diesem Vektor werden die Komponenten für den Merkmalsvektor bestimmt, der sich aus den SPs des Vektors ϕ und den Eigenvektoren $u_1, ..., u_K$ zusammensetzt. Dafür wird für alle $i = 1, ..., K$

$$\omega_i = \phi_1 \cdot (u_i)_1 + ... + \phi_n \cdot (u_i)_n$$

berechnet. Daraus resultiert für jedes Bild ein K-Zeilen Merkmalsvektor

$$\Omega = (\omega_1, ..., \omega_K)$$

welcher zum Abgleich verwendet wird. Durch die ED des Merkmalsvektors von X und denen der Datenbank $Y_1, ..., Y_n$ sowie einem Schwellwert kann dann entschieden werden, ob zwei Bilder das gleiche Gesicht zeigen.

Ähnliche Algorithmen

Es gibt zwei weiter Gesichtserkennungs-Algorithmen die auf der Arbeit von Eigenfaces aufbauen, zum Einen die Fisherfaces und zum Anderen die Laplacianfaces. Im Folgenden werden die Grundlagen der beiden Algorithmen vorgestellt.

Fisherfaces Die von Belhumeur et al. 1997 eingeführten Fisherfaces [BHK97] sind dem Eigenfaces-Algorithmus sehr ähnlich. Lediglich die Basisvektoren für den Subraum werden anders ausgewählt. Statt der PCA zur Bestimmung der Eigenvektoren und Eigenwerte wird die Fisher's Linear Discriminant (FLD) verwendet. Daher heißen die Eigenvektoren des Subraums Fisherfaces analog zur Bezeichnung bei Eigenfaces. Diese Methode erreicht eine höhere Streuung der Klassen im Subraum als PCA. Dadurch ist diese Methode unempfindlicher gegenüber wechselnden Lichtverhältnissen und verschieden Gesichtsausdrücken und führt auch allgemein zu genaueren Ergenissen. Die Merkmalsextraktion von Bildern und der privtatsphäre-schützende Abgleich kann auf die gleiche Weise wie bei Eigenfaces stattfinden.

Laplacianfaces Laplacianfaces [HYH+05] wurden von Xiaofei He et al. 2005 vorgestellt. Auch in diesem Algorithmus ist der Unterschied zu Eigenfaces die Auswahl der Basisvektoren zur Repräsentation der Merkmalsvektoren. Statt PCA oder FLD wird Locality Preserving Projections (LPP) verwendet. Dabei wird ein locality preserving face Subraum aus den daraus entstehenden Eigenvektoren gebildet, welche Laplacianfaces genannt werden. Dieser Subraum versucht die innere Geometrie der Bilder und die lokale Struktur zu erfassen. Auch hier findet die Merkmalsextraktion und der privtatsphäre-schützende Abgleich auf die gleiche Weise wie bei Eigenfaces statt. Die Genauigkeit übersteigt die von Fisherfaces und damit auch die der Eigenfaces.

2.5.2 SCiFI

Das System for Secure Face Identification [OPJM10] wurde 2010 von Osadchy et al. an der Universität Haifa entwickelt. Für dieses System wurde ein Gesichtserkennungs-Algortihmus entwickelt, der im Vergleich zu den anderen vorgestellten Algorithmen speziell für die Anforderungen der sicheren Berechnung des Abgleichs entwickelt wurde. Das System basiert auf der Berechnung von Hamming-Distanzen, welche effizienter sicher berechnet werden können als euklidische Distanzen. Dafür wird jedes Bild als 900 bit Vektor dargestellt, unabhängig der tatsächlichen Größe des Bildes. Diese Merkmalsvektoren setzten sich aus mehreren Komponenten zusammen, die im Folgenden beschrieben werden. Für den Abgleich ist weitaus weniger Aufwand als bei Eigenfaces nötig, da hier einfach die HD der Merkmalsvektoren berechnet werden müssen. Die niedrigste Distanz, sofern diese kleiner als ein bestimmter Schwellwert ist, wird als Ergebnis zurückgegeben.

Merkmalsextraktion

Die Merkmalsvektoren haben eine feste Länge von 900 bit. Diese 900 bit setzen sich aus räumlichen und Erscheinungs Komponenten zusammen. Um die Komponenten für die Vektoren zu erhalten, muss der Algorithmus die Position von fünf Schlüsselmerkmalen kennen: die Nase, beide Augen und beide Mundwinkel. Dafür können Methoden wie [OLM07; VJ01] zum Erkennen von Gesichtsmerkmalen zum Einsatz kommen.
Die folgenden Vorberechnungen müssen vorgenommen werden, um mit dem Algorithmus Merkmalsvektoren zu erhalten. Dafür ist eine Menge an Bildern nötig, die keinen Bezug zu den später abgeglichen Bildern hat. Für diese Bilder wurden 34 3D Modelle aus der USF HumanID 3D Face Database[1] gerendert, sodass die Gesichter auf den Bildern mittig zentriert und frontal zu sehen sind.
Aus diesen Bildern werden anschließend p Patches extrahiert. Ein Patch steht für einen Bildausschnitt, durch welche später die Bilder repräsentiert werden. Diese Patches werden von jedem Bild für jeden Bildausschnitt gespeichert und indiziert, um als Vokabular mit N Worten für die Erscheinungs-Komponente zu dienen. Außerdem wird die räumliche Komponente gespeichert, in welchem der Abstand der Mitte der Patches zur Gesichtsmitte gespeichert wird. Diese Werte werden in Q Bins aufgeteilt, welche indiziert sind.
Ein neues Bild kann als Vektor repräsentiert werden, indem für jeden Patch p die Indizes der n ähnlichsten Patches aus N möglichen Worten für diesen Gesichtsausschnitt ausgesucht werden. Ebenso wird bei der räumlichen Komponente verfahren. Hier werden die Indizes der z nächsten diskreten Distanzwerte aus den Q möglichen gespeichert. Diese Indizes werden in binärer Form repräsentiert, wodurch sich $p \cdot (N+Q)$ bits für den Merkmalsvektoren ergeben. [OPJM10] schlägt die Nutzung von $p = 30$ Patches mit einem Vokabluar von $N = 20$ vor, aus denen die $n = 4$ ähnlichsten ausgwählt werden, und $Q = 10$ Distanzen aus denen die $z = 2$ nächsten ausgewählt werden. Daraus ergibt sich ein $30 \cdot (20+10) = 900$bit Vektor. Die ersten 10 Bit pro Patch stehen für den räumlichen Komponenenten bei dem 2 Bits auf 1 gesetzt sind. Die 20 weiteren Bits stehen für die Erscheinungs-Komponent, bei welchem 4 Bits auf 1 gesetzt sind. Somit haben die Merkmalsvektoren ein konstantes Hammingewicht von $30 \cdot (2+4) = 180$. Diese Merkmalsvektoren können dann über die XOR Operation miteinander verglichen werden. Je ähnlicher zwei Bilder sind, desto niedriger ist die HD. Wenn zwei Bilder identisch sind, ergibt sich eine HD von 0, da alle gesetzten Bits an gleicher Stelle sind.

[1] USF HumanID 3D Face Database, Courtesy of Prof. Sudeep Sarkar, University of South Florida, Tampa, FL.

3 Verwandte Arbeiten

Im folgenden Kapitel werden verschiedene Arbeiten zum Thema privatsphäre-schützende Gesichtserkennung vorgestellt. Dafür muss der Abgleich der Vektoren auf eine sichere Art erfolgen, wodurch der Server keine Informationen über das Bild des Client erhält und der Server seine Datenbank geheim hält. Dafür wird der Abgleich durch S2C-Protkolle berechnet. Das S2C-Protokoll kann unabhängig vom Algorithmus gewählt werden und somit gibt es verschiedenste privatsphäre-schützende Systeme der beiden Gesichtserkennungs-Algorithmen Eigenfaces und SCiFI. Die folgenden Algorithmen und Frameworks sind sicher gegenüber passiven Angreifern und es wird angenommen, dass die Kommunikation vor allem über einen sicheren Kanal stattfindet.

3.1 Privacy-Preserving Face Recognition [EFG+09]

In [EFG+09] zeigen Erkin et al. wie man Eigenfaces für S2C basierend auf HE erweitern kann. Dafür müssen sowohl die Bilder verschlüsselt übertragen werden, als auch der Abgleich auf eine Art stattfinden, die sowohl das Bild des Clients X, als auch die Datenbank Y des Servers geheim hält. Da der verwendete Eigenfaces-Algorithmus für die Projektion des Bildes X in den Merkmalsvektorraum eine Kommunikation mit dem Server erfordert, wird hier S2C eingesetzt. Auch hier kann der Algorithmus in drei Phasen aufgeteilt werden. Als erstes verschlüsselt der Client sein Bild X und schickt die Ciphertexte der einzelnen Pixel $Enc(x_1), ..., Enc(x_n)$ an den Server. Dieser verschlüsselt die Werte des Durchschnittsvektors $-\psi$ und kann dann durch die additiv homomorphe Eigenschaft der Verschlüsselung die Subtraktion

$$Enc(x_i - \psi_i) = Enc(x_i) \cdot Enc(-\psi_i) = Enc(\phi_i)$$

durchführen. Da der Server die Werte $u_i, ..., u_K$ im Plaintext kennt, kann er die SPs

$$Enc(\omega_i) = Enc(\phi_1)^{(u_i)_1} \cdot ... \cdot Enc(\phi_n)^{(u_i)_n}$$

mit allen Eigenfaces berechnen. So ergibt sich $\Omega = \omega_1, ..., \omega_K$ als Merkmalsvektor.

Als zweites muss der Server die verschlüsselten Distanzen $ED(\Omega, \Omega')$ des Merkmalsvektor Ω aus dem ersten Schritt mit allen Merkmalsvektoren Ω' der Datenbank Y berechnen. Als Distanzmaß wird die quadratische euklidische Distanz benutzt, welche der Server aber nur durch Kommunikation mit dem Client berechnen kann. Die ED kann als

$$ED = \sum_{i=1}^{K} \omega_i^2 + \sum_{i=1}^{K} (-2\omega_i \omega'_i) + \sum_{i=1}^{K} \omega'^2_i$$

formuliert werden. Da der Server nur Ω' im Plaintext kennt, kann er nur den zweiten und dritten Summanden der Gleichung berechnen. Den zweiten Summanden kann er berechnen, indem er

$$Enc(\sum_{i=1}^{K} (-2\omega_i \omega'_i)) = \prod_{i=1}^{K} Enc(\omega_i)^{-2\omega'_i}$$

setzt. Für den ersten Summanden ist die Interaktion mit dem Client nötig, da für die Distanz das Quadrat der Client-Eingabe benötigt wird und diese nicht mittels homomorpher Eigenschaften auf den verschlüsselten Werten berechnet werden kann. Um diese Quadrierung zu ermöglichen und gleichzeitig die Werte der Merkmalsvektoren geheim zu halten, maskiert der Server die verschlüsselten Werte mit Zufallszahlen r_i und schickt $Enc(\omega_i + r_i)$ für alle $i = 1, ..., K$ an den Client. Dieser entschlüsselt die Werte und berechnet

$$S = \sum_{i=1}^{K} (\omega_i + r_i)^2$$

um anschließend $Enc(S)$ verschlüsselt an den Server zu übertragen. Der Server berechnet daraufhin

$$Enc(\sum_{i=1}^{K} \omega_i^2) = Enc(S) \cdot \prod_{i=1}^{K} (Enc(\omega_i)^{-2r_i} \cdot Enc(-r_i^2)) \quad .$$

Dieser Wert kann für alle Abgleiche mit den Merkmalsvektoren Ω' der Datenbank Y verwendet werden, da er ausschließlich durch die Client-Eingabe X bestimmt wird. Mit diesen Werten kann der Server alle Distanzen $ED(\Omega, \Omega')$ berechnen.

Als letzten Schritt muss noch das Minimum gefunden und mit einem Schwellwert verglichen werden. Um aus M verschlüsselten Distanzen das Minimum zu bestimmen, werden durch eine rekursive Prozedur jeweils zwei Distanzen durch ein kryptographisches Protokoll verglichen, welche eine randomisierte Version der niedrigeren Distanz zurück gibt. Nach dem am Ende der Rekursion nur eine, nämlich die minimale Distanz übrig bleibt, kann diese auf die gleiche Weise mit einem Schwellwert verglichen werden. Der daraus resultierende Wert kann entweder ein Index sein, also der Name der Person auf einem Bild, oder eine binäre Ausgabe. Zum Schluss muss der Server diesen verschlüsselten Wert noch an den Client schicken, da nur dieser ihn entschlüsseln kann.

3.2 Efficient Privacy-Preserving Face Recognition [SSW10]

Die in 3.1 vorgestellte Arbeit wurde von Sadegh et al. in [SSW10] durch einen hybriden Ansatz der S2C effizienter gemacht. Dabei werden die Projektions- und die Distanz-Phase

der Eigenfaces Berechnung nach wie vor homomorph durchgeführt, die abschließende Minimumsberechnung wird jedoch mit einem GC-Protokoll durchgeführt. Außerdem kann die Distanz-Phase durch das Zusammenpacken der Ciphertexte vom Server zum Client optimiert werden, wodurch die Kommunikation verbessert wird. Dadurch wurde nicht nur die benötigte Zeit zur Berechnung kürzer, sondern auch die Kommunikation der beiden Parteien effizienter. Im Vergleich zur vorherigen Arbeit [EFG+09], welche $O(\log N)$ bei N Datenbankeinträgen benötigt, wird bei dem hybriden Protokoll nur noch eine konstante Anzahl an Runden benötigt.

Um ein hybrides Protokoll zu ermöglichen, müssen die homomorph verschlüsselten Werte in garbled Werte konvertiert werden. Dafür wird ein ParalellConvert-Protokoll [KSS09] verwendet. Dabei maskiert der Server seine verschlüsselten Distanzwerte mit zufälligen Werten und schickt diese an den Client. Um die Maskierung aufzuheben, führt dieser dann mit den entschlüsselten maskierten Werten zusammen mit dem Server ein GC-Protokoll aus, woraus der Client die garbled Werte erhält. Mit diesen Werten wird dann der Minimum Circuit aufgestellt, welcher als Output den geringsten Wert und den entsprechenden Namen liefert. Das hybride Protokoll verbessert die online Kommunikation der Minimum-Phase um bis zu Faktor 15 bei kurzen Sicherheitsparametern, das Packen der Ciphertexte der Distanz-Phase verbessert diese Phase um bis zu Faktor 4. Des Weiteren wurde ein komplett auf GC basierendes Protokoll entwickelt, welches jedoch ineffizienter als das hybride Protokoll ist, vor allem aufgrund der hohen offline Kommunikation für den gesamten GC.

3.3 SCiFI – A System for Secure Face Identification [OPJM10]

Die Grundlagen und Merkmalsextraktion des SCiFI Systems wurden schon in 2.5.2 beschrieben. An dieser Stelle stellen wir noch die privatsphäre-schützende Berechnung des Systems vor. Auch hier soll wieder sowohl das Bild des Clients X, als auch die Datenbank Y mit M Einträgen des Servers geheim gehalten werden. Im Vergleich zu Eigenfaces wird hier die Merkmalsextraktion nicht mit berücksichtigt. Das ist jedoch unproblematisch, da bei SCiFI beide Parteien ihren Algorithmus auf einer für beide Seiten bekannten Datenbank anlernen können und somit die Vektoren auf gleicher Basis erstellen und dementsprechend vergleichbar sind. Eine solche Lösung für die Erstellung der Merkmalsvektoren könnte auch Eigenfaces und darauf aufbauende Algorithmen effizienter machen.
Das ursprüngliche System von Osadchy et al. [OPJM10] nutzt additive HE. Dabei verschlüsselt der Client seinen $\ell = 900$ bit Merkmalsvektor X und schickt ihn an den Server mit der Datenbank Y. Der Server kann dann durch die homomorphe Eigenschaft die Hamming Distanzen aller Vektoren berechnen, indem er für den Vektor $X = \{x_0, ..., x_{\ell-1}\}$ und alle Vektoren Y^i aus $Y^1, ..., Y^M$ mit $Y^i = \{y_0^i, ..., y_{\ell-1}^i\}$ das XOR

$$Enc(x_j \oplus y_j^i) = Enc(x_j) \cdot (1 - y_j^i) + (1 - Enc(x_j)) \cdot y_j^i$$

und anschließend die Summe

$$Enc(HD(X, Y^i)) = \sum_0^{\ell-1} Enc(x_j \oplus y_j^i)$$

berechnet. Der Server wählt dann M zufällige Zahlen $r^i \mod 181$ und berechnet für jeden Vektor Y^i

$$Enc(HD(X, Y^i) + r^i).$$

Diese Ciphertexte schickt der Server an den Client, der diese entschlüsseln kann. Die entschlüsselten Werte $\mod 181$ nimmt der Client als Auswahlwerte für M Aufrufe einer OT_1^{181} mit dem Server, der als Eingabe jeweils $U_0, ..., U_{181}$ hat, wobei

$$U_j = \begin{cases} 1 & \text{if } 0 \leq (j - r^i) \mod 181 \leq t \\ 0 & \text{sonst} \end{cases}.$$

Der Client erhält als Ergebnis genau dann eine 1, wenn ein Merkmalsvektor in der Datenbank eine kleinere HD zu dem eigenen Merkmalsvektor als ein gegebener Schwellwert t aufweist. Das Protokoll liefert also für jeden Vektor in der Datenbank eine 0 oder 1. Es kann auch um eine Minimumsfunktionalität ergänzt werden, damit wie bei Eigenfaces auch nur der minimale Wert als Ergebnis zurück geliefert wird.

3.4 Frameworks zur sicheren Zwei-Parteien Berechnung

In diesem Abschnitt werden bisherige Arbeiten zu Frameworks der S2C kurz vorgestellt, die für verschiedene biometrische Erkennungsalgorithmen verwendet werden können.

3.4.1 TASTY: Tool for Automating Secure Two-partY computations [HöS+10]

Das 2010 eingeführte TASTY [HöS+10] ermöglicht effiziente S2C ausgehend von einer high-level Sprache, TASTYL gennant, zu generieren. Dafür kann der Benutzer sowohl HE als auch Yao's GC sowie eine hybride Lösung aus beiden Ansätzen nutzen. Der Compiler generiert anschließend aus einer Folge von Operationen und Instruktionen ein möglichst effizientes Protokoll. Die meisten Berechnungen werden dafür in die eher zeit-unkritische Setup-Phase gelegt, um eine effiziente Online-Phase zu ermöglichen. Dadurch konnte TASTY vorhergehende Tools, wie z.B. Fairplay [BNP08], welches in Java implementiert ist, sowohl in Zeit- als auch Kommunikations-Effizienz übertreffen. Beispielsweise bei der Berechnung der AES-Funktionalität mit ultra-short Security Parameter konnte die Berechnungszeit um knapp 17,5% und die Kommunikation um 85% verbessert werden. Das Tool kann außerdem zum Benchmarking und Vergleichen von Protokollen eingesetzt werden.

3.4.2 GMW vs. Yao? Efficient Secure Two-Party Computation with Low Depth Circuits [SZ13]

In [SZ13] optimierten Schneider et al. das GMW-Protokoll und zeigten, dass ein optimiertes GMW-Protokoll effizienter sein kann, als state-of-the-art GC-Implementierungen.
GMW erfordert pro AND-Gatter im BC eine Interaktion der beiden Parteien in der Online-Phase, wodurch die Kommunikation linear mit der Tiefe des BC wächst. Dadurch hängt die Leistung von GMW sehr stark von der Netzwerklatenz ab. Um dies zu verbessern, versucht man BCs zu generieren, die sowohl eine geringe Größe als auch eine niedrige Tiefe aufweisen, wodurch weniger Kommunikationsrunden notwendig sind.
Um einen BC mit geringerer Tiefe zu erreichen und gleichzeitig die Größe nicht wesentlich zu erhöhen, wurden spezielle Building-Blocks verwendet, wie z.B. der Ladner-Fischer-Adder [LF80]. Um das Protokoll schneller zu machen, ersetzten MTs die vorgerechneten OTs zur Auswertung der ADD-Gatter, wodurch die Online-Phase nochmal verbessert werden konnte. Außerdem wurde AES statt SHA-1 zur Erzeugung von pseudo Zufallszahlen verwendet.
Durch diese Optimierungen konnte das neue GMW-Protokoll die Berechnungszeiten für die privatsphäre-schützende Gesichtserkennung von Eigenfaces im Vergleich zu 3.2 und 3.4.1 um mindestens Faktor 8 verbessern. Bei SCiFI konnte im Vergleich zu 3.3 die Setup-Phase im lokalen Netzwerk bei 100 Bildern von 213s auf 0.3s und die Online-Phase von 31s auf 0.006s verbessert werden.

3.4.3 GSHADE: Faster Privacy-Preserving Distance Computation and Biometric Identification [BCF+14]

GSHADE [BCF+14] hat das in [BCP13] vorgestellte SHADE-Protokoll als Grundlage genommen, was für Secure Hamming Distance Computation steht, und basiert wie dieses nur auf OT. Es verallgemeinert dieses Protokoll, daher heißt es generalized SHADE, um auch andere Distanz-Metriken effizient und sicher zu berechnen.
Um die Kommunikations-Komplexität zu optimieren, basiert das Protokoll auf C-OT. Da es bei der biometrischen Identifikation um den Abgleich von einer Eingabe eines Cients mit N Eingaben eines Servers geht, profitiert man von dem Fakt, dass die Zahl der OTs gleich bleibt und nur die Strings für den OT bei größerem N länger werden. Dadurch konnte es die bis dahin schnellste Lösung zur Berechnung von Eigenfaces und SCiFI, das in 3.4.2 vorgestellte GMW-Protokoll, deutlich unterbieten. Bei Eigenfaces konnte ein Speedup um Faktor 20 erreicht werden, bei SCiFI immerhin Faktor 4 bis 5.

3.5 Vergleich von Eigenfaces and SCiFI

Im Vergleich zu Eigenfaces wurde SCiFI speziell für effiziente S2C entworfen. Dabei sollten keine Abstriche in der Genauigkeit der Gesichtserkennung gemacht werden. Um diese Ziele zu erreichen wurde ein komplett neues System entwickelt, welches einige Vorzüge

gegenüber bestehenden Systemen wie Eigenfaces aufweist. Es musste eine Repräsentation von Bildern gefunden werden, die effizient sicher abgeglichen werden können. Bestehende Gesichtserkennungs-Algorithmen beruhten auf stetigen Werten, welche nicht effizient sicher berechenbar waren. Eine Abbildung der stetigen Werte auf diskrete Werte führt zu geringerer Genauigkeit in der Gesichtserkennung. Daher wurde für SCiFI eine Repräsentation gewählt, die unabhängig von der Bildauflösung ein Gesicht durch stetige, konkret binäre Werte in einem immer gleich großen Vektor repräsentiert. Das macht den privatsphäre-schützenden Abgleich besonders effizient und da er bei großen Datenbanken die meiste Zeit beansprucht, ist somit das gesamte System schneller, als bestehende Systeme, die wie Eigenfaces auf der euklidischen Distanz beruhen. Abgesehen von der privatsphäre-schützenden Berechnung sind aber auch die Erkennungsraten von SCiFI besser als die von Eigenfaces.
Zum Vergleich wurden in [OPJM10] die Datenbanken Carnegie Mellon University–Pose, Illumination, and Expression Database [SBB03] (CMU-PIE) und Face Recognition Technology Database [PMRR00] (FERET) verwendet. Dabei dient die CMU-PIE Datenbank dazu, die Genauigkeit des Algorithmus bei Veränderungen in der Beleuchtung der Gesichter zu testen. Die FERET Datenbank spiegelt realistischere Bilder wieder, mit leichten Abweichungen in Gesichtsausdruck und Pose. Anhand dieser wurde ermittelt, das SCiFI bei CMU-PIE zu 80% die korrekte Person als ersten Treffer zurückliefert, und zu 95% ist der korrekte Treffe in den ersten 6 Ergebnissen. Bei FERET wird zu 65% die richtige Person an erster Stelle identifiziert und zu 95% befindet sich diese in den ersten 20 Treffern.
Bei der FERET Datenbank [PMRR00] mit einer false positive Rate von 5% konnte SCiFI zu 97% die richtige Person identifizieren, wohingegen der Eigenfaces Algorithmus bei gleicher false positiv Rate nur zu 87% die korrekt Person erkennt. Bei der CMU-PIE Datenbank [SBB03] fällt der Unterschied noch größer aus, da SCiFI bei einer false positiv Rate von 15% zu 93% richtig identifiziert und Eigenfaces nur zu 46%.
Ein weiterer Vorteil von SCiFI ist, dass es die hohen Erkennungsraten mit nur einem Bild pro Person in der Datenbank erreicht. Das ist für ein reales Szenario, bei dem von einer gesuchten Person eventuell nur ein Bild vorliegt, ein großer Vorteil gegenüber Eigenfaces, welches mit mehreren Bildern angelernt werden muss. Aus diesen Gründen wurde die Merkmalsextraktion von SCiFI für den Demonstrator ausgewählt. Diese ist bereits in JAVA 1.6 and Matlab implementiert.

4 Sichere Zwei-Parteien Berechnung mit ABY

S2C ermöglicht zwei Parteien eine beliebige Funktion zu berechnen und dabei nur das Ergebnis der jeweils anderen Partei offen zu legen. Für unseren Demonstrator haben wir das im Folgenden vorgestellte ABY-Framework ausgewählt, welches ein etabliertes Open-Source Framework[1] ist. Es ermöglicht eine effiziente S2C mit state-of-the-art Optimierungen und mixed-protocol Berechnungen. Dabei gilt wieder die Sicherheit gegenüber passiven Angreifern.

4.1 Das ABY-Framework

Das mixed-protocol Framework ABY erlaubt die Kombination von drei Protokollen zur S2C, siehe 2.3, basierend auf Arithmetic Sharing, Boolean Sharing mit GMW und Yao Sharing. Da die Protokolle jeweils bei verschiedenen Operationen effizientere Repräsentationen haben, wie Arithmetic Circuits bei Multiplikation und BC bei bitweisen Vergleichen, kann eine Kombination von Protokollen zu einem deutlichen Performanzgewinn führen. Das Framework erlaubt auch Nicht-Experten S2C durchzuführen, da das Framework von den S2C-Protokollen abstrahiert. Man kann sich dies wie eine Virtuelle Maschine vorstellen, die unabhängig von der Hardware läuft. Die Protokolle nutzen jeweils state-of-the-art Optimierungen um bestmögliche Ergebnisse zu erreichen. Die zu berechnende Funktion wird dafür bei Boolean Sharing und Yao Sharing als BC oder als Arithmetic Circuit bei Arithmetic Sharing dargestellt. Die Eingaben der beiden Parteien werden entweder als Shares bei Arithmetic und Boolean Sharing oder als Keys bei Yao's Sharing repräsentiert. Die Sharings unterstützen Standardoperationen wie Addition, Multiplikation und bitweise Vergleiche. Die unterschiedlichen Sharing-Typen benutzen verschiedene Protokolle, um diese Operationen auszuführen. Arithmetic Sharing nutzt Beaver's MTs [Bea91], Boolean Sharings nutzt GMW [GMW87] und Yao's Sharing nutzt GC [Yao86].
ABY ist nicht das erste Tool zur Gestaltung von mixed-protocols, aber bisherige Ansätze wie [HöS+10], welche HE für Arithmetic Ciruits und ein GC-Protokoll für BC verwendeten, brachten nur geringe Laufzeitverbesserungen da die Konversionen nicht effizient berechnet werden konnten und somit die Vorteile wieder ausglichen. Die Konversion bei ABY zwischen den Sharing-Typen und somit auch dem Protokoll wird durch effiziente vorberechnete OT-Erweiterungen ermöglicht. Um eine effiziente Onlinephase zu gewährleisten werden die meisten kryptographischen Funktionen in der Setup-Phase vorberechnet.

[1] `http://encrypto.de/code/ABY`

Das Framework ermöglicht Konversionen nur in bestimmte Richtungen. Für unsere Zwecke sind jedoch keine Konversionen nötig, da die minimale HD alleine mit Yao's Sharing/Boolean Sharing effizient berechnet werden kann.

Das Framework ist in C/C++ implementiert und ermöglicht die Angabe eines zu evaluierenden Circuits in dieser Hochsprache. Dafür wird ein Circuit angelegt, der entweder vom Typ Boolean Circuit oder Arithmetic Circuit sein kann. In diesem Circuit werden Gatter, die verschiedene Standardoperationen ausführen können, durch Wires verbunden, auf welchen Werte anliegen. Die Eingaben der beiden Parteien in den Circuit werden durch Secret Sharing als Share an das erste Gatter übergeben. Der gesamte Circuit wird dann auf Shares ausgewertet. Je nach Sharing Typ, Arithmetic, Boolean oder Yao, werden dafür die verschiedenen Protokolle verwendet. Am Ende des Circuits führen die Parteien eine Rekombination der Shares durch, um den Plaintext der Ausgabe zu erhalten. Dabei kann ausgewählt werden, welche Partei das Ergebnis erhalten soll.

4.2 Minimum Hamming Distance Circuit

Um die privatsphäre-schützende Berechnung von SCiFI zu ermöglichen nutzen wir ABY, um die minimale HD zwischen Client und Server zu berechnen. Dafür legen wir einen BC an, der diese Funktionalität repräsentiert. Dieser kann anschließend durch Yao Sharing mit dem GC-Protokoll oder durch Boolean Sharing mit dem GMW-Protokoll ausgewertet werden. Das Framework verwendet für das GC-Protokoll die free-XOR [KS08], point-and-permute [BNP08], und garbled row-reduction[NPS99; PSSW09] Optimierungen sowie die größenoptimierte Generierung von Circuits aus [KSS09]. Für das GMW-Protokoll wird die tiefenoptimierte Erstellung von Circuits aus [SZ13] und die Vorbrechnung der MTs durch R-OT aus [ALSZ13] verwendet.

Der BC bekommt als Eingabe die Merkmalsvektoren des Servers sowie die Namen zu den entsprechenden Vektoren. Ebenso wird der Merkmalsvektor des Clients und der clientseitig individuell einstellbaren Schwellwert übergeben. Außerdem werden Variablen initialisiert, die in der Schleife zur Berechnung aller HDs benötigt werden, siehe Code-Ausschnitt 4.1.

Auflistung 4.1: Minimum Hamming Distance Circuit 1/3

```
1 /** share Array S is servers DB of feature vectors,
2  *  share C is Client feature vector,
3  *  share Array N is db names to corresponding feature vectors,
4  *  share NM contains empty String,
5  *  share T is Threshold set by the client,
6  *  uint32_t n ist db size, uint32_t d is dimension of feature
       vectors,
7  *  bc is the circuit where we put the gates on.
8  */
9 share** build_min_hamming_dist_circuit(share** S, share* C, share
      ** N, share* T,
```

```
10          share* NM, uint32_t n, uint32_t d, BooleanCircuit* bc) {
11
12      uint32_t i, j, h;
13      share **mindist, **ids, **distance, **ge;
14
15      distance = (share**) malloc(sizeof(share*) * n);
16      mindist = (share**) malloc(sizeof(share*) * n);
17      ids = (share**) malloc(sizeof(share*) * n);
18      ge = (share**) malloc(sizeof(share*) * 2);
19
20      //variables for PutWideAddGate for calculating the sum of the
            XORs
21      //builds tree structure for efficient computation of sums
22      uint32_t resbitlen = ceil_log2(d);
23      vector<vector<uint32_t> > ins(d);
24
25      //represents constant zero value
26      uint32_t zerogate = bc->PutConstantGate(0);
```

Nach der Initialisierung werden in einer Schleife das bitweise XOR aller Datenbankeinträge des Servers mit dem Vektor des Clients berechnet und die Summe der Einsen in den resultierenden Bitstrings berechnet. Das Ergebnis sind die HDs, welche in shares gespeichert werden. siehe Code-Ausschnitt 4.2.

Auflistung 4.2: Minimum Hamming Distance Circuit 2/3

```
27      //loop through db elements and compute hamming distance to
            client input
28      for (h=0; h < n; h++) {
29          //compute XOR for db elements and client input
30          S[h] = bc-> PutXORGate(S[h], C);
31          for(i = 0; i<d; i++){
32              //tree structure for PutWideAddGate
33              //needs 2D-array of size ins[d][ceil(log_2(d))]
34              ins[i].resize(resbitlen);
35              for(j = 0; j<resbitlen; j++){
36                  if(j == 0){
37                      //set XOR bit
38                      ins[i][j] = S[h]-> get_wire(i);
39                  }
40                  else{
41                      //set 0 bit
42                      ins[i][j] = zerogate;
43                  }
44              }
45          }
46          //add all bits of the XOR value
47          distance[h] = new boolshare(bc->PutWideAddGate(ins), bc);
48      }
```

Der Code-Ausschnitt 4.3 zeigt, wie aus den HDs zusammen mit den entsprechenden Namen das Minimum ausgewählt wird. Dieses Minimum wird anschließend mit dem Schwellwert verglichen. Falls die HD kleiner als der Schwellwert ist, wird der entsprechende Name als Ergebnis gespeichert, ansonsten wird ein leerer String gespeichert. Der String, welcher als share vorliegt, wird dann als Ergebnis des Circuits übergeben und kann von den Parteien ausgewertet werden.

Auflistung 4.3: Minimum Hamming Distance Circuit 3/3

```
49      //get minimum hamming distance and corresponding name
50      //result is stored in mindist (distance value) and ids (name)
51      bc->PutMinIdxGate(distance, N, n, mindist, ids);
52
53      //output 1, if mindist is smaller than threshold, else 0
54      ge[0] = bc->PutGEGate(T, mindist[0]);
55
56      //output corresponding name if mindist is smaller than
            threshold,
57      //else nm (empty string)
58      ge[1] = bc->PutMUXGate(ids[0], NM, ge[0]);
59
60      //free variables
61      for(i = 0; i < n; i++) {
62          free(distance[i]);
63      }
64      free(mindist);
65      free(distance);
66      free(ids);
67
68      //return output share to parties
69      return ge;
70  }
```

5 Demonstrator

Im abschließenden Kapitel führen wir den neuen Demonstrator ein, der auf dem in C/C++ implementierten ABY-Framework und dem in Java implementierten SCiFI basiert. Dabei nutzen wir ABY zur privatsphäre-schützenden Berechnung der HD und SCiFI als Grundlage für das neue GUI und zur Merkmalsextraktion, welche in Matlab implementiert ist.

5.1 Dokumentation

Das GUI basiert auf dem vorhandenen SCiFI System und wurde optimiert und ergänzt. Es ist in ein Server- und ein Client-Programm unterteilt, welche auf entfernten Rechnern ausgeführt werden. Beide Parteien müssen sich am Anfang auf einen Port und die Ausgabe einigen, clientseitig muss auch die IP-Adresse des Servers eingegeben werden, siehe Abbildung 5.1.

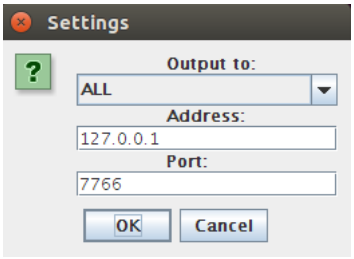

Abbildung 5.1: Settings

Die Kommunikation zwischen beiden Parteien findet ausschließlich über den sicheren Minimum Hamming Distance Circuit (MHDC) aus 4.2 und dem zugrundeliegenden ABY-Framework statt. Nach Auswahl der Netzwerkeinstellungen wird der MHDC-Prozess gestartet und beide Parteien erhalten Rückmeldung sobald der Verbindungsaufbau erfolgreich war. Um dies zu ermöglichen, müssen der MHDC-Prozess und das GUI kommunizieren. Da nur der Verbindungsaufbau, der individuelle Schwellwert und nach Abgleich das Ergebnis ausgetauscht werden müssen, konnte die Kommunikation über eine einfache STD I/O gelöst werden.
Der MHDC wurde als executable in die Struktur des SCiFI Systems integriert, um auf die Merkmalsvektoren zugreifen zu können. Das GUI muss dem Circuit-Prozess nun lediglich mitteilen, dass ein neuer Abgleich stattfinden soll. Als Parameter wird dabei der clientseitige

Schwellwert übergeben, welcher für jeden Abgleich neu gewählt werden kann. Anschließend liest der Prozess entweder serverseitig die Datenbank oder clientseitig einen einzelnen Merkmalsvektor ein. Nach Ausführen des MHDC wird das Ergebnis der vorher festgelegten Partei oder beiden Parteien per STD I/O mitgeteilt. Dadurch kann das Ergebnis über das GUI angezeigt werden.

Abbildung 5.2 zeigt das Client User Interface, welches erlaubt ein Bild von einer Webcam oder aus einer Datei auszuwählen sowie den Schwellwert für den Abgleich festzulegen. Wenn ein Bild für den Abgleich ausgewählt wird, muss aus diesem zunächst ein entsprechender Merkmalsvektor generiert werden. Dafür muss das Gesicht mit seinen fünf Schlüsselmerkmalen im Bild erkannt werden. Dies geschieht in der vorliegenden Implementierung durch die Viola-Jones-Methode [VJ01] automatisch, die Position der Schlüsselmerkmale kann jedoch bei Bedarf noch manuell angepasst werden. Falls der Client das Ergebnis erhält, erscheint dieses in Form eines Popups.

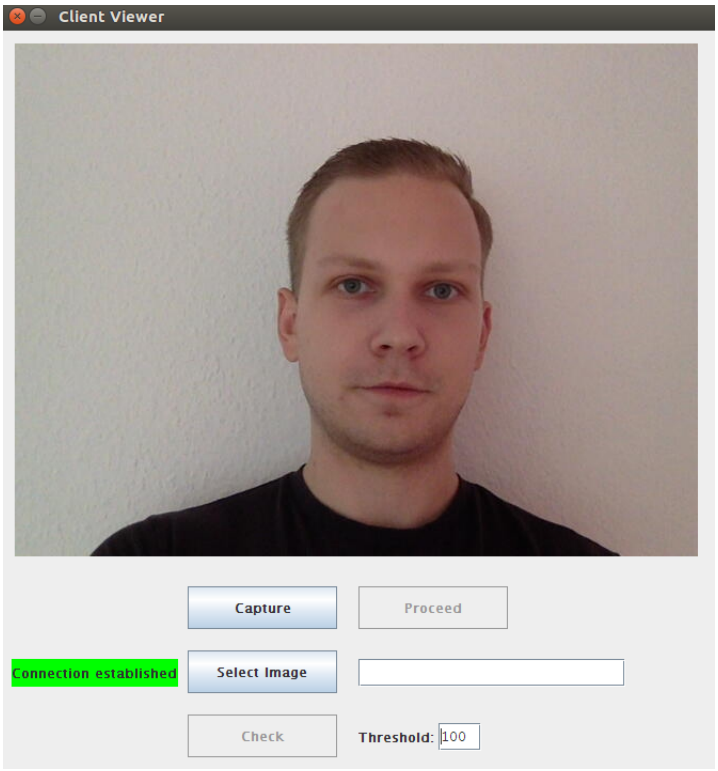

Abbildung 5.2: Client UI

Im Vergleich zum ursprünglichen SCiFI System wurde das Server User Interface um die Möglichkeit der Datenbank-Verwaltung ergänzt, siehe Abbildung 5.3. Während der Server läuft, können also Bilder hinzugefügt und gelöscht werden, ohne den Prozess beenden zu müssen. Neben dem Log des Servers, in dem der Verbindungsstatus und je nach Einstellung die Ergebnisse des Abgleichs angezeigt werden, sieht man alle aktuell in der Datenbank

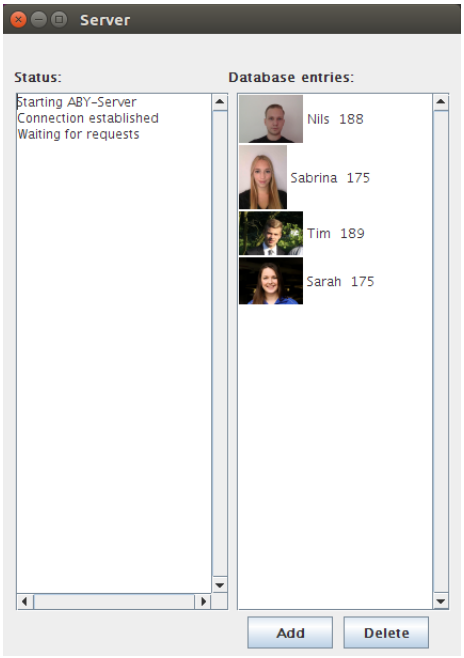

Abbildung 5.3: Server UI

vorhandenen Bilder mit Namen und individuellem Schwellwert. Dieser Schwellwert ist die minimale HD zu den anderen Bildern der Datenbank. Da der Schwellwert clientseitig gesetzt werden kann, dient der serverseitig berechnete Schwellwert nur zur Information der Unterscheidbarkeit von Bildern. Wenn man ein Bild hinzufügen möchte, hat man auch hier die Wahl, ob man es mit einer Webcam aufnimmt oder eine vorhandene Bilddatei auswählt. Das Bild wird wie auch beim Abgleich in einen Merkmalsvektor umgewandelt und anschließend in die Datenbank geschrieben.

5.2 Evaluation und Performance-Vergleich

Die S2C wurde durch das Ersetzten der HE mit dem MHDC, welcher im ABY Framework umgesetzt wurde und entweder mit Yao's GC-Protokoll oder dem GMW-Protokoll evaluiert werden kann, deutlich effizienter. Die Merkmalsextraktion in Matlab wurde jedoch nicht optimiert, wodurch sie, wie im originalen SCiFI System, konstant circa 8 Sekunden bei einer Auflösung von 640x480 Pixeln pro Bild benötigt. Eine Umsetzung in C/C++ könnte laut [OPJM10] zu einer Verbesserung der Laufzeit um Faktor 4 bis 5 führen, sodass die Merkmalsextraktion in unter 2 Sekunden durchgeführt werden könnte. Die Erkennung der Schlüsselmerkmale wurde ebenfalls beibehalten. Sie benötigt für ein Bild 1,5 Sekunden zur Erkennung des Gesichts und Markierung der 5 Schlüsselmerkmale bei oben genannter Auflösung.

	$n = 100$				$n = 320$				$n = 1000$			
	Lokal		Internet		Lokal		Internet		Lokal		Internet	
κ in bit	80	128	80	128	80	128	80	128	80	128	80	128
Yao's GC	1,5	1,6	2,1	2,5	5,0	5,2	6,9	7,9	15,5	17,4	20,0	23,5
GMW	**1,3**	**1,3**	**2,0**	**2,2**	**4,4**	**4,5**	**5,5**	**5,7**	**13,4**	**13,7**	**14,5**	**15,6**

Tabelle 5.1: Laufzeit in Sekunden für den MHDC für verschiedene Datenbankgrößen im LAN- und Internet-Setting mit sym. Sicherheitsparameter $\kappa = 80$ bit und $\kappa = 128$ bit.

	$n = 100$		$n = 320$		$n = 1000$	
κ in bit	80	128	80	128	80	128
Yao's GC	17,5	28,1	56	89,7	175,1	280,1
GMW	**8,6**	**13,6**	**27,7**	**42,9**	**86,7**	**193,1**

Tabelle 5.2: Kommunikation in MB für den MHDC für verschiedene Datenbankgrößen mit sym. Sicherheitsparameter $\kappa = 80$ bit und $\kappa = 128$ bit.

Um die Zeit für einen Abgleich mit dem neuen Demonstrator in einem realistischen Szenario nachzustellen, wurden die Messungen auf zwei entfernten Amazon EC2 `m4.large` Instanzen ausgeführt. Diese sind mit 64-bit Intel Xeon dualcore CPUs mit 2.4 GHz und 8 GB RAM ausgestattet. Die Server-Instanz wird in Frankfurt ausgeführt und die Client-Instanz in Irland. Die Verbindungsgeschwindigkeit liegt bei 424 Mbit/s und die Latenz bei 23 ms. Für das LAN-Setting wurden zwei der `m4.large` Instanzen in Frankfurt ausgeführt. Alle Ergebnisse sind die Durchschnittswerte von jeweils 10 Messungen. Um die Ergebnisse mit ähnlichen Arbeiten vergleichen zu können, benutzen wir short-term security Parameter, also für den symmetrischen Sicherheitsparameter $\kappa = 80$ bit und für den asymmetrischen $\rho = 1024$ bit. Da diese Parameter aber nur bis 2010 vom NIST [NIS12] empfohlen wurden, haben wir auch die Laufzeiten für long-term security Parameter $\kappa = 128$ und $\rho = 3072$ gemessen, welche bis mindestens 2030 verwendet werden können.

Tabelle 5.1 zeigt die benötigten Zeiten für einen Abgleich durch dem MHDC mit Yao's GC-Protokoll und mit dem GMW-Protokoll bei verschiedenen Datenbankgrößen im oben beschriebenen Internet-Setting und im LAN-Setting mit Gigabit-LAN. Dabei ist das GMW-Protokoll in allen getesteten Einstellungen dem Yao's GC-Protokoll überlegen. In Tabelle 5.2 zeigen die Ergebnisse, dass das GMW-Protokoll konstant weniger als die Hälfte der Kommunikation von Yao's GC-Protokoll benötigt.
Tabelle 5.3 zeigt einen Vergleich des neuen Demonstrators zu dem ursprünglichen SCiFI-System und einem nicht-privatsphäre-schützenden Abgleich auf einer Datenbank mit $n = 100$ Einträgen. Dabei setzt sich die Laufzeit von SCiFI aus konstanten 213s für die Setup-Phase und 0,31s pro Bild in der Datenbank für die Online-Phase zusammen, bei $n = 100$ entspricht das $100 \cdot 0,31 = 31$s. Der Abgleich in ABY besteht aus 0,2s für die Setup-Phase und 1,1s für die Online-Phase, wobei beide Phasen abhängig von der Datenbankgröße sind. Da die Merkmalsextraktion identisch ist, wurden die Zeiten dafür nicht berücksichtigt. Die Ergebnisse

Protokoll	**ABY (GMW)**	SCiFI (HE)	SCiFI (Unverschlüsselt)
Programmiersprache	C++	Java	Java
Zeit in Sek.	**1,3**	244	0,005
Kommunikation in MB	8,6	**7,3**	0,001

Tabelle 5.3: Vergleich des neuen Demonstrators mit dem ursprünglichen SCiFI und einem unverschlüsselten Abgleich von HDs bei Datenbankgröße $n = 100$.

zeigen, dass der neue Demonstrator zum Abgleich nur $0,53\%$ der Zeit des ursprünglichen Systems benötigt, was einer Verbesserung um Faktor 188 entspricht. Der Vergleich mit dem ungeschützten System dient dazu aufzuzeigen, welche Laufzeit-Einbußen für den Privatsphärenschutz hingenommen werden müssen. Dabei schickt der Client seinen Merkmalsvektor unverschlüsselt an den Server, damit dieser die minimale HD berechnen kann und teilt dem Client anschließend das Ergebnis mit. Ein ungeschützter Abgleich benötigt so nur 1ms zur Berechnung des minimalen HD sowie 4ms zur Kommunikation. Der privatsphäre-schützende Abgleich mit ABY benötigt also etwa 260 mal so lange, wie der ungeschützte Abgleich von Merkmalsvektoren. Ebenso könnte die Datenbank auch direkt bei dem Client gespeichert werden, wodurch keine Kommunikation mehr nötig wäre.

5.3 Fazit

Der neue Demonstrator ist im Vergleich zu SCiFI, dem bisher einzigen Demonstrator für privatsphäre-schützende Gesichtserkennung, eine Verbesserung sowohl in der Laufzeit, als auch der Benutzerfreundlichkeit. Mit der wesentlich schnelleren Berechnung und der Datenbankverwaltung ist er anwendbar für reale Szenarien, wie z.B bei Grenzkontrollen, Überwachung von öffentlichen Plätzen oder Eingangskontrollen von Unternehmen.
Die bisherige Merkmalsextraktion in Matlab ist noch nicht die effizienteste Lösung. Eine zukünftige Umsetzung der Merkmalsextraktion in C/C++ könnte es aber ermöglichen, den Demonstrator für Echtzeit Videoüberwachung anwendbar zu machen. Dafür würde jedes erkannte Gesicht in einem Video automatisch die Merkmalsextraktion auslösen und anschließend ein Abgleich mit der Server-Datenbank durchgeführt werden. Wenn ein solches System das Vertrauen der Öffentlichkeit erlangt hat, könnte die Gesichtserkennung an kritischen Orten ausgebaut werden und somit die Sicherheit erhöhen.
Es sei aber auch angemerkt, dass die Videoüberwachung mit Gesichtserkennung als Sicherheitsmerkmal ihre Grenzen hat, da es nötig ist, zumindest einen Großteil des Gesichts zu erkennen. Gegenüber verschiedensten Kopfbedeckungen wie Motorradhelmen, religiöse Verschleierungen etc. ist auch die Gesichtserkennung wirkungslos.

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

BC Boolean Circuit

C-OT Correlated OT

CMU-PIE Carnegie Mellon University pose, illumination, and expression

ED Euklidische Distanz

FERET Face Recognition Technology

FLD Fisher's Linear Discriminant

GC Garbled Circuits

GMW Goldreich-Micali-Wigderson

HD Hamming-Distanz

HE Homomorphic Encryption / Homomorphe Verschlüsselung

LPP Locality Preserving Projections

MHDC Minimum Hamming Distance Circuit

MT Multiplication Triple

OT Oblivious Transfer

PCA Principal Component Analysis

R-OT Random OT

S2C Secure Two-Party Computation / Sichere Zwei-Parteien Berechnung

SP Skalarprodukt

Literatur

[ALSZ13] G. ASHAROV, Y. LINDELL, T. SCHNEIDER, M. ZOHNER. **"More efficient oblivious transfer and extensions for faster secure computation"**. In: *Computer and Communications Security (CCS)*. ACM, 2013, S. 535–548 (siehe S. 7, 21).

[BCF+14] J. BRINGER, H. CHABANNE, M. FAVRE, A. PATEY, T. SCHNEIDER, M. ZOHNER. **"GSHADE: Faster Privacy-preserving Distance Computation and Biometric Identification"**. In: *Workshop on Applied Homomorphic Cryptography (WAHC)*. ACM, 2014, S. 187–198 (siehe S. 18).

[BCP13] J. BRINGER, H. CHABANNE, A. PATEY. **"SHADE: Secure HAmming DistancE computation from oblivious transfer"**. In: *Workshop on Information Hiding and Multimedia Security (IH&MMSec'14)*. Springer, 2013, S. 164–176 (siehe S. 18).

[Bea91] D. BEAVER. **"Efficient multiparty protocols using circuit randomization"**. In: *CRYPTO'91*. Springer, 1991, S. 420–432 (siehe S. 7, 20).

[BHK97] P. N. BELHUMEUR, J. P. HESPANHA, D. J. KRIEGMAN. **"Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection"**. In: *IEEE Transactions Pattern Analysis and Machine Intelligence*. 1997, S. 711–720 (siehe S. 12).

[BNP08] A. BEN-DAVID, N. NISAN, B. PINKAS. **"FairplayMP: a system for secure multi-party computation"**. In: *ACM Conference on Computer and Communications Security (ACM CCS'08)*. ACM, 2008, S. 257–266 (siehe S. 17, 21).

[CHK+12] S. G. CHOI, K.-W. HWANG, J. KATZ, T. MALKIN, D. RUBENSTEIN. **"Secure Multi-Party Computation of Boolean Circuits with Applications to Privacy in On-Line Marketplaces"**. In: *Topics in Cryptology*. Springer, 2012, S. 416–432 (siehe S. 10).

[DGK08] I. DAMGÅRD, M. GEISLER, M. KRØIGAARD. **"Homomorphic encryption and secure comparison"**. In: *International Journal of Applied Cryptography*. 2008, S. 22–31 (siehe S. 10).

[DSZ15] D. DEMMLER, T. SCHNEIDER, M. ZOHNER. **"ABY - a framework for efficient mixed-protocol secure two-party computation"**. In: *In Network and Distributed System Security (NDSS'15)*. The Internet Society, 2015, S. 239–254 (siehe S. 3).

[EFG+09] Z. ERKIN, M. FRANZ, J. GUAJARDO, S. KATZENBEISSER, I. LAGENDIJK, T. TOFT. **"Privacy-Preserving Face Recognition"**. In: *Privacy Enhancing Technologies Symposium (PETS'09)*. Springer, 2009, S. 235–253 (siehe S. 11, 14, 16).

[GMW87] O. GOLDREICH, S. MICALI, A. WIGDERSON. **"How to Play ANY Mental Game"**. In: *ACM Symposium on Theory of Computing (ACM STOC '87)*. ACM, 1987, S. 218–229 (siehe S. 2, 9, 20).

[HEKM11] Y. HUANG, D. EVANS, J. KATZ, L. MALKA. **"Faster secure two-party computation using garbled circuits"**. In: *USENIX Security Symposium*. USENIX Association, 2011 (siehe S. 9).

[HöS+10] W. HENECKA, S. K. ÖGL, A.-R. SADEGH, T. SCHNEIDER, I. WEHRENBERG. **"TASTY: Tool for Automating Secure Two-party Computations"**. In: *ACM CCS'10*. ACM, 2010, S. 451–462 (siehe S. 17, 20).

[HYH+05] X. HE, S. YAN, Y. HU, P. NIYOGI, H.-J. ZHANG. **"Face recognition using Laplacianfaces"**. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2005), S. 328–340 (siehe S. 12).

[IKNP03] Y. ISHAI, J. KILIAN, K. NISSIM, E. PETRANK. **"Extending oblivious transfers efficiently"**. In: *CRYPTO*. Springer, 2003, S. 145–161 (siehe S. 7).

[KS08] V. KOLESNIKOV, T. SCHNEIDER. **"International Colloquium on Automata, Languages and Programming (ICALP)"**. In: *Improved garbled circuit: Free XOR gates and applications*. Springer, 2008, S. 486–498 (siehe S. 9, 21).

[KSS09] V. KOLESNIKOV, A.-R. SADEGHI, T. SCHNEIDER. **"Improved garbled circuit building blocks and applications to auctions and computing minima"**. In: *Cryptology and Network Security (CANS '09)*. Springer, 2009 (siehe S. 16, 21).

[LF80] R. LADNER, M. FISCHER. **"Parallel prefix computation"**. In: *Journal of the ACM 27(4)*. 1980, S. 831–838 (siehe S. 18).

[NIS12] NIST. *NIST Special Publication 800-57, Recommendation for Key Management Part 1: General (Rev. 3)*. Techn. Ber. 2012 (siehe S. 27).

[NPS99] M. NAOR, B. PINKAS, R. SUMNER. **"Privacy preserving auctions and mechanism design"**. In: *Electronic Commerce (EC'99)*. ACM, 1999, S. 129–139 (siehe S. 9, 21).

[OLM07] M. OSADCHY, Y. LECUN, M. MILLER. **"Synergistic face detection and pose estimation with energy-based models"**. In: *Journal of Machine Learning Research*. 2007, S. 1197–1215 (siehe S. 13).

[OPJM10] M. OSADCHY, B. PINKAS, A. JARROUS, B. MOSKOVICH. **"SCiFI - A System for Secure Face Identification"**. In: *IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2010, S. 239–254 (siehe S. 2, 12 f., 16, 19, 26).

[Pai99] P. PAILLIER. **"Public-key cryptosystems based on composite degree residuosity classes"**. In: *EUROCRYPT*. Springer, 1999, S. 223–238 (siehe S. 10).

[PMRR00] P. J. PHILLIPS, H. MOON, S. A. RIZVI, P. J. RAUSS. **"The FERET evaluation methodology for face-recognition algorithms"**. In: *IEEE Transactions Pattern Analysis and Machine Intelligence*. 2000, S. 1090–1104 (siehe S. 19).

[PSSW09] B. PINKAS, T. SCHNEIDER, N. P. SMART, S. C. WILLIAMS. **"Secure two- party computation is practical"**. In: *ASIACRYPT'09*. Springer, 2009, S. 250–267 (siehe S. 9, 21).

[Rab81] M. O. RABIN. *How to exchange secrets with oblivious transfer*. 1981 (siehe S. 7).

[RAD78] R. RIVEST, L. ADLEMAN, M. DERTOUZOS. **"On data banks and privacy homomorphism"**. In: *Foundations of Secure Computation*. 1978, S. 168–177 (siehe S. 2, 10).

[SBB03] T. SIM, S. BAKER, M. BSAT. **"The CMU pose, illumination, and expression database"**. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2003, S. 1615–1618 (siehe S. 19).

[SSW10] A.-R. SADEGHI, T. SCHNEIDER, I. WEHRENBERG. **"Efficient Privacy-Preserving Face Recognition"**. In: *12th International Conference on Information Security and Cryptology (ICISC '09)*. Springer, 2010, S. 229–244 (siehe S. 15).

[SZ13] T. SCHNEIDER, M. ZOHNER. **"GMW vs. Yao? Efficient Secure Two-Party Computation with Low Depth Circuits"**. In: *Financial Cryptography and Data Security (FC'13)*. Springer, 2013, S. 275–292 (siehe S. 18, 21).

[TP91] M. TURK, A. PENTLAND. **"Eigenfaces for recognition"**. In: *Journal of Cognitive Neuroscience*. 1991, S. 71–86 (siehe S. 2, 10).

[VJ01] P. VIOLA, M. JONES. **"Rapid object detection using a boosted cascade of simple features"**. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*. 2001, S. 511–518 (siehe S. 13, 25).

[Yao86] A. C. YAO. **"How to Generate and Exchange Secrets"**. In: *Foundations of Computer Science (FOCS '86)*. IEEE Computer Society, 1986, S. 162–167 (siehe S. 2, 8, 20).