Karlsruher Institut für Technologie

**Fakultät für Informatik**
**Institut für Anthropomatik und Robotik**
**Lehrstuhl für Intelligente Sensor-Aktor-Systeme (ISAS)**
Prof. Dr.-Ing. Uwe D. Hanebeck

Intelligente ISAS Sensor-Aktor-Systeme

# Masterarbeit

## Datenschutzerhaltende Zustandsschätzung mittels eines kryptographisch sicheren Kalman-Filterverfahrens

**Mikhail Aristov**

31. Juli 2018

Referent: Prof. Dr.-Ing. Uwe D. Hanebeck

Betreuer: Dr.-Ing. Benjamin Noack

**Zusammenfassung**

Sichere Signalverarbeitung befasst sich mit den Fragestellungen der verteilten Verarbeitung von vertraulichen Messungen und Systemzustände. Dabei werden formale Methoden der modernen Kryptographie in den Verfahren der Signalverarbeitung, z. B. in dem Kalman-Filter, eingesetzt, um ihre Berechnung beweisbar sicher zu gestalten. Ihre Anwendungsgebiete reichen von der verteilten Filterung in Sensornetzwerken bis zur ausgelagerten Berechnung in der Cloud.

Der erste Beitrag dieser Arbeit ist ein Abriss über die essentiellen Definitionen und Werkzeuge der formalen Kryptographie, der speziell auf das Feld der sicheren Signalverarbeitung mittels homomorpher Verschlüsselung zugeschnitten ist. Damit analysieren wir ein aus der Literatur bekanntes Verfahren zur verteilten Berechnung des Extended Kalman-Filters und zeigen, wieso es die intuitive Sicherheitsdefinition nicht erfüllt. Unsere Hauptbeiträge sind zwei neue Verfahren zur verteilten Datenfusion und Filterung in Sensornetzwerken mithilfe der vertraulichen Informations- und Konsensfilter. Für beide legen wir vollständige Sicherheitsbeweise und Komplexitätsanalysen vor und setzen sie prototypisch um, um ihre Genauigkeit mithilfe numerischer Simulationen auszuwerten. Am Schluss präsentieren wir eine Auswahl der offenen Fragestellungen der sicheren Signalverarbeitung, die als ein Leitfaden für die zukünftige Recherche auf dem Gebiet dienen kann. *

* An dieser Stelle möchte ich mich herzlich bei Jeremias Mechler von der Arbeitsgruppe Kryptographie und Sicherheit des Instituts für Theoretische Informatik (ITI) für sein umfangreiches Feedback und vertiefende Diskussionen bedanken, ohne die diese Arbeit nicht in der Form zustande gekommen wäre.

# Eidesstattliche Erklärung

Hiermit erkläre ich, die vorliegende Masterarbeit selbstständig angefertigt zu haben. Die verwendeten Quellen sind im Text gekennzeichnet und im Literaturverzeichnis aufgeführt.

Karlsruhe, 31. Juli 2018

Mikhail Aristov

# Inhaltsverzeichnis

**Abbildungsverzeichnis** III

**Tabellenverzeichnis** V

**Notation** VII

**1 Einleitung** 1

**2 Stand der Forschung** 3
2.1 Signalverarbeitung . . . 3
2.1.1 Kalman-Filter . . . 5
2.1.2 Informationsfilter . . . 7
2.1.3 Konsensfilter . . . 8
2.2 Kryptographie . . . 11
2.2.1 Asymptotische Sicherheit . . . 11
2.2.2 Homomorphe Verschlüsselung . . . 12
2.2.3 Sichere Mehrparteienberechnung . . . 19
2.3 Quantisierung und Fixpunktarithmetik . . . 24
2.4 Aktuelle Herausforderungen . . . 26

**3 Sicheres Extended Kalman-Filter** 29
3.1 Protokoll . . . 29
3.2 Rundungsfehleranalyse . . . 32
3.3 Sicherheitsmodell . . . 34
3.4 Sicherheitsanalyse . . . 35
3.5 Bewertung . . . 37

**4 Sicheres verteiltes Informationsfilter** 39
4.1 Sicherheitsmodell . . . 40
4.2 Protokoll . . . 41
4.3 Sicherheitsbeweis . . . 43
4.3.1 Korrektheit . . . 43

4.3.2 Vertraulichkeit . . . . . 44
4.4 Komplexitätsanalyse . . . . . 51
4.5 Erweiterungen des Grundprotokolls . . . . . 53
4.5.1 Zusätzliche Hubs . . . . . 53
4.5.2 Normalisieren des Informationsvektors . . . . . 54
4.6 Genauigkeitsauswertung mittels numerischer Simulation . . . . . 61
4.7 Bewertung . . . . . 65

**5 Sicheres Gossip-Konsensfilter 69**
5.1 Sicherheitsmodelle . . . . . 69
5.2 Sicheres Gossip-Konsenfilter mit reellen Eingaben . . . . . 72
5.2.1 Protokoll . . . . . 72
5.2.2 Sicherheitsbeweis . . . . . 72
5.3 Sicheres Gossip-Konsenfilter in endlichen Körpern . . . . . 78
5.3.1 Protokoll . . . . . 78
5.3.2 Korrektheit . . . . . 80
5.3.3 Vertraulichkeit . . . . . 80
5.4 Komplexitätsanalyse . . . . . 81
5.5 Genauigkeitsauswertung mittels numerischer Simulation . . . . . 82
5.6 Bewertung . . . . . 84

**6 Ergebnisse 87**

# Abbildungsverzeichnis

4.1 Aufbau des sicheren verteilten Informationsfilter . . . . . 43
4.2 Aufbau der Simulaiton des sicheren verteilten Informationsfilters . . . 62

# Tabellenverzeichnis

2.1 Übersicht der Abkürzungen für homomorphe Operationen . . . . . 14

4.1 Ergebnisse der Evaluation des sicheren verteilten Informationsfilters . 64

5.1 Ergebnisse der Evaluation des sicheren Gossip-Konsensfilters . . . . . 84

# Notation

## Konventionen

| | |
|---|---|
| $x$ | Skalar |
| $\boldsymbol{x}$ | Zufallsvariable |
| $\hat{x}$ | Erwartungswert der Zufallsvariable $\boldsymbol{x}$. |
| $\underline{x}$ | Spaltenvektor |
| $\underline{\boldsymbol{x}}$ | Zufallsvektor |
| $\underline{\hat{x}}$ | Erwartungswert des Zufallsvektors $\underline{\boldsymbol{x}}$. |
| $\mathbf{A}$ | Matrix |
| $(.)_k$ | Quantität zum Zeitpunkt $k$. |
| $\mathbb{R}$ | Menge der reellen Zahlen. |
| $\mathbb{N}$ | Menge der natürlichen Zahlen (ohne Null). |
| $\square$ | Ende eines Beweises. |
| $[\![.]\!]$ | Chiffrat; Kurzschreibweise für $\mathsf{Enc}(.)$. |
| $\equiv$ | Äquivalenzrelation. |
| $\overset{\text{c}}{\equiv}$ | Komplexitätstheoretische Ununterscheidbarkeit. |
| $:=$ | Deterministische Zuweisung. |
| $\leftarrow$ | Probabilistische Zuweisung, z.B. $(pk, sk) \leftarrow \mathsf{KeyGen}(1^n)$. |
| | Uniformes Samplen aus einer Menge, z.B. $b \leftarrow \{0, 1\}$. |

## Abkürzungen

| | |
|---|---|
| (E)KF | (Extended) Kalman Filter |
| IF | Information Filter |
| PPT | Probabilistic Polynomial Time |
| RMSE | Root Mean Square Error |

KAPITEL 1

# Einleitung

Die vorliegende Arbeit befasst sich mit der Problematik der beweisbar sicheren verteilten Zustandsschätzung und Datenfusion. Während das Thema Sicherheit in den Informationstechnologien immer mehr Bedeutung in den letzten Jahren gewonnen hat, wurde es auf dem Forschungsgebiet der Signalverarbeitung vor allem ab 2013 immer präsenter. Die aktuelle Forschung befasst sich intensiv damit, die historische Kluft zwischen den Gebieten der Signalverarbeitung und der Kryptographie zu überbrücken, jedoch fällt es beiden Seiten oft schwer, die Hintergründe und Motivationen der anderen nachzuvollziehen. Das führt dazu, dass vermeintlich sichere Werkzeuge und Verfahren entwickelt werden, für die sich entweder keine formale Sicherheitsdefinition, oder kein praktisches Einsatzszenario finden lassen.

Die erste Aufgabe dieser Arbeit ist es, die theoretischen Grundlagen der sicheren Signalverarbeitung zusammenzufassen. Damit schaffen wir die Basis für die Entwicklung von verteilten Filterverfahren, die sowohl beweisbar sicher, als auch praktisch einsetzbar sind. Weil wir nicht alle Filterverfahren und nicht alle Sicherheitsmodelle hier behandeln können, fokussieren wir uns auf die praktische Umsetzung des Kalman-Filters und zwei seiner Varianten, nämlich Informations- und Konsensfilter, im sog. „semiehrlichen“ Sicherheitsmodell, das in der Kryptographie als Basis für stärkere Sicherheitsdefinitionen dient. Die dazu notwendigen Präliminarien aus der Kryptographie sind u. a. die homomorphen Verschlüsselungsschemas, sichere Mehrparteienberechnung, Protokollkomposition und Eingabenquantisierung.

Unser erster Beitrag ist die Sicherheitsanalyse eines bereits existierenden Verfahrens zur verteilten Berechnung des Extended Kalman Filters mit verschlüsselten Messungen. Anhand dieses Beispiels zeigen wir die Gefahren des *ad hoc* Protokollentwurfs ohne eine übergreifende Sicherheitsdefinition auf, die wir bei unseren eigenen Beiträgen beachten werden.

Unsere neuen Protokolle setzen das Informationsfilter und das Gossip-Konsensfilter für verteilte Sensornetzwerke um. Für beide stellen wir basierend auf den sys-

temtheoretischen Grundlagen ein geeignetes Sicherheitsmodell auf und führen einen formalen Sicherheitsbeweis durch, sowie eine Komplexitätsanalyse bzgl. ihres Rechen- und Netzwerkaufwands, um ihre Skalierbarkeit für reale Anwendung einzuschätzen. Für beide erstellen wir prototypische Umsetzungen und werten anhand dieser die Genauigkeit der Schätzungen im jeweiligen Protokoll mit unterschiedlichen Quantisierungseinstellungen aus.

Zum Schluss listen wir weitere Schritte auf, die notwendig sind, um mit unseren Verfahren stärkere Sicherheitsgarantien zu erreichen, sowie relevante Alternativansätze für die zukünftige Forschung.

KAPITEL 2

# Stand der Forschung

In diesem Kapitel geben wir die Übersicht der wichtigsten Verfahren aus der Signalverarbeitung und der modernen Kryptographie. Unser Ziel dabei ist, die Grundlagen und die Tools zu erklären, die wir benötigen, um die Fragestellungen der beweisbar sicheren Signalverarbeitung kryptographisch korrekt anzugehen und die Sicherheit unserer Verfahren formal zu belegen. Außerdem verweisen wir auf relevante Studien, die für die zukünftige Forschung auf dem Gebiet wichtig sein können.

Im Abschnitt 2.1 stellen wir kurz die drei Filterverfahren vor, für die wir in den Kapiteln 3, 4 und 5 kryptographisch sichere Protokolle analysieren bzw. entwerfen wollen: das Kalman-Filter, das Informationsfilter und das Konsensfilter. Im Abschnitt 2.2 erklären wir die Konzepte und die Werkzeuge der modernen Kryptographie, die wir in unseren Protokollen bzw. in unseren Sicherheitsbeweisen verwenden: die IND-CPA-Sicherheit, homomorphe Verschlüsselungen (am Beispiel des Paillier-Schemas), die Angreifermodelle im Simulationsparadigma, und die sichere Protokollkomposition. Im Abschnitt 2.3 bringen wir die Gebiete der Signalverarbeitung und der Kryptographie über die Quantisierung und die Fixkommaarithmetik letztendlich zusammen.

## 2.1 Signalverarbeitung

Unter Signalverarbeitung verstehen wir die praktische Anwendung der Signal- und Systemtheorie, die sich „mit der Beschreibung von Signalen und mit der Beschreibung, Analyse und Synthese von Systemen“ befassen. Ihre primären Einsatzgebiete liegen in der Nachrichten-, Regelungs- und Messtechnik [57].

*Signale* sind physikalische Größen, die Energie und Information über Phänomene übertragen und somit die physikalische Repräsentation des Wissens darüber darstellen. Das Umwandeln der Signale in verarbeitbare Form (*Messung*) ist die Aufgabe der Sensorik. Allgemeine Signalverarbeitung geht oft von harmonischen (sinusförmigen) Signalen aus, die oft mathematisch als Funktionen im Zeitbereich (Differenzi-

algleichungen, Stossantwort) oder im Bildbereich (Übertragungsfunktion, Frequenzgang) modelliert werden, wobei beide Formen ineinander mittels der Laplace- bzw. Fourier-Transformation konvertierbar sind.

Ein *System* in unserem Sinne ist jede Anordnung, die Signale (also Energie bzw. Information) verarbeitet. Ein allgemeines System hat Eingangsgrößen bzw. -signale und wandelt diese in Ausgangsgrößen (-signale) um. Genauso wie die Signale selbst, modellieren wir diese Umwandlung mathematisch als einen Satz von Gleichungen, der das System charakterisiert. Weil kein Modell aber die Realität vollständig abbilden kann, berücksichtigen wir darin grundsätzlich nur die relevanten Eigenschaften des Systems, um die Beschreibung sinnvoll einfach zu halten.

Systeme werden nach ihren Eigenschaften klassifiziert. *Statische* und *dynamische Systeme* unterscheiden sich darin, dass die letzteren einen internen Zustand haben, der sich über die Zeit verändern kann. Systeme können außerdem kontinuierlich oder diskret bzgl. der Zeit und des Zustandsraums, linear oder nichtlinear, sowie zeitinvariant oder zeitvariabel in ihren Gleichungen sein [40]. Lineare zeitinvariante Systeme (engl. *linear time-invariant*, kurz: LTI) sind in der Signalverarbeitung wegen ihrer Superpositionseigenschaften ein besonders wichtiger Spezialfall [57].

**Zustandsschätzung** In vielen Situationen werden die Parameter und der interne Zustand eines dynamischen Systems gesucht, die nicht direkt messbar sind. Stattdessen sind oft nur die verrauschten Messungen seiner Ein- und Ausgaben, sowie sein mathematisches Modell gegeben. Ein *Systemmodell* besteht aus vier Komponenten: den deterministischen Beschreibungen seiner Zustandsentwicklung in der Zeit (*Transitionsmodell*) und der Abbildung von dem Zustand auf die Ausgaben (*Messmodell*), sowie den stochastischen Rauschmodelle für den Zustand und die Messungen.

Ein Transitionsmodell ist im Allgemeinen nichtlinear, oft aber als ein lineares differentiales Gleichungssystem darstellbar, wenn wir davon ausgehen können, dass der jeweils nächste Zustand nur von dem aktuellen und der Systemeingabe abhängt (die *Markov-Annahme*). Beim Messmodell ist oft die Eigenschaft der „Beobachtbarkeit“ (engl. *observability*) gefragt, die bezeichnet, informell, ob ein Systemzustand aus den erhobenen Messdaten komplett berechenbar ist. Es existieren formale Kriterien zum Nachweisen der Beobachtbarkeit, diese sind aber für LTI-Systeme grundsätzlich einfacher zu berechnen, als für andere Kategorien [40, Abschn. 2.3.7].

Die deterministischen Beschreibungen werden um stochastische Modelle des Zustands- und Messrauschens ergänzt, weil, wie bereits erwähnt wurde, kein Modell perfekt ist. Um Einflüsse zu berücksichtigen, die entweder zu aufwendig zu messen oder gar nicht messbar sind, fügen wir in das Systemmodell stochastische Zufallsvariablen ein

und treffen sinnvolle Annahmen über ihre Verteilungen.

Eine Methode zum Berechnen des nicht direkt erfassbaren Zustands bzw. der Parameter eines Systems aus seinem Modell und den verrauschten Ein- und Ausgaben bezeichnen wir als ein „Schätzer“ (engl. *estimator*). Schätzverfahren unterteilen sich in Filterung[1] (Schätzung des aktuellen Zustands mit allen Messungen bis zum aktuellen Zeitpunkt), Prädiktion (Schätzung eines zukünftigen Zustands) und Glättung (Schätzung eines Zustands mit Messungen, die zeitlich danach erhoben wurden) [40].

Von besonderem Interesse sind die sog. „rekursiven Schätzer“, in denen die aktuelle Schätzung den gesamten Messverlauf zusammenfasst, der dadurch nicht zusätzlich gespeichert werden muss. In den Unterabschnitten 2.1.1, 2.1.2 und 2.1.3 werden drei rekursive Schätzer vorgestellt, die für die verteilte Zustandsschätzung besonders wichtig sind: das Kalman-Filter, das Informationsfilter und das Konsensfilter [61].

**Sichere Signalverarbeitung** In den letzten Jahren werden immer mehr Arbeiten veröffentlicht, die die Ansätze zur *vertraulichen* verteilten Zustandsschätzung mit den Methoden der modernen Kryptographie untersuchen (siehe die Übersichtsstudien [2, 3, 49, 75]). Diese können grob in die allgemeinen Grundbausteine (die bereichsübergreifend anwendbar sind) und konkrete Applikationen aus der Nachrichten- und Regelungstechnik unterteilt werden.

Zu den grundlegenden kryptographischen Bausteinen für die vertrauliche Signalverarbeitung zählen z. B. der sichere Zahlenvergleich [72], Gram-Schmidtsches Orthogonalisierungsverfahren [27], diskrete Kosinusberechnung [12], diskrete Fourier-Transformation [11], *Nearest-Neighbour*-Verfahren [66], $k$-means Clustering [71, 44, 43, 15], Median-Berechnung und Table-Lookup [41, Kap. 8–9].

Konkrete vertrauliche Verfahren für praktische Anforderungen schließen ein: Gesichtserkennung [23, 68], Inhaltsempfehlung über kollaboratives Filtern [25], EKG-Monitoring [7, 8], DNS-Matching [70, 74], biometrische Abgleiche der Fingerabdrücke [6] und der Irisaufnahmen [54], Fingerprinting und Wasserzeichendetektion für DRM [10], Gesundheitsmonitoring über Smartphones [51], Wasserzeichen in Videostreams [13] und intelligente Stromversorgung (engl. *smart grid*) [24].

### 2.1.1 Kalman-Filter

Das Kalman-Filterverfahren (KF) wurde 1960 von Rudolf E. Kalman veröffentlicht [47] und ist eine Methode, um aus verrauschten und teilweise redundanten

1 In der Signalverarbeitung wird der Begriff „Filter“ allgemein für die Rekonstruktion des Systemzustands aus den verrauschten Messungen verwendet, z. B. Bandbreitenfilter. Informell werden auch gesamte Schätzverfahren oft nach dem darin verwendeten Filter benannt.

Messungen den nicht direkt erfassbaren Zustand rekursiv abzuschätzen [55]. Im Folgenden betrachten wir nur Systemmodelle, in denen gilt:

$$\underline{\boldsymbol{x}}_{k+1} = \mathbf{A}_k \underline{\boldsymbol{x}}_k + \mathbf{B}_k \underline{\boldsymbol{u}}_k$$
$$\underline{\boldsymbol{z}}_k = \mathbf{H}_k \underline{\boldsymbol{x}}_k + \underline{\boldsymbol{v}}_k,$$

wobei $\underline{\boldsymbol{x}}_k$ ein Zufallsvektor, der den internen Systemzustand der Länge $L$ zum Zeitpunkt $k$ repräsentiert[2], $\mathbf{A}_k$ die Transitionsmatrix, $\mathbf{B}_k$ die Steuerungsmatrix ($\mathbf{A}_k, \mathbf{B}_k$ bilden zusammen das lineare Transitionsmodell), $\underline{\boldsymbol{u}}_k$ der Steuerungsvektor (Eingabesignal), $\underline{\boldsymbol{z}}_k$ der Messvektor (Ausgabe), $\mathbf{H}_k$ die Messmatrix (lineare Messmodell) und $\underline{\boldsymbol{v}}_k$ das Messrauschvektor sind [60].

Unter den Annahmen, dass der Fehler der Schätzung und das Systemrauschen, sowie der Schätzfehler und das Messrauschen jeweils unkorreliert und das Messrauschen und der Schätzfehler mittelwertfrei sind, können wir dann ein Schätzverfahren aufstellen, das aus dem Prädiktionsschritt und dem Filterschritt besteht. Im ersten Schritt prädizieren wir den Erwartungswert des Systemzustands $\underline{\hat{x}}^{\mathrm{p}}_{k+1}$ und seine Unsicherheit als Kovarianzmatrix $\mathbf{C}^{\mathrm{p}}_{k+1}$ zum Zeitpunkt $k+1$ als Funktion von der Schätzung $\underline{\hat{x}}^{\mathrm{e}}_k, \mathbf{C}^{\mathrm{e}}_k$, dem Eingabesignal $\underline{\hat{u}}_k$ und dem Systemrauschen $\mathbf{C}^w_k$ zum Zeitpunkt $k$ [61]:

$$\underline{\hat{x}}^{\mathrm{p}}_{k+1} = \mathbf{A}_k \underline{\hat{x}}^{\mathrm{e}}_k + \mathbf{B}_k \underline{\hat{u}}_k \tag{2.1a}$$
$$\mathbf{C}^{\mathrm{p}}_{k+1} = \mathbf{A}_k \mathbf{C}^{\mathrm{e}}_k \mathbf{A}^{\mathrm{T}}_k + \mathbf{C}^w_k. \tag{2.1b}$$

Im Filterschritt wollen wir unsere Prädiktion $\underline{\hat{x}}^{\mathrm{p}}_k$ mit den Informationen aus der Messung $\underline{\hat{z}}_k$ verbessern, indem wir, intuitiv, aus dem tatsächlichen und dem erwarteten Messwert ein *Korrekturterm* ($\underline{\hat{z}}_k - \mathbf{H}_k \underline{\hat{x}}^{\mathrm{p}}_k$) berechnen, ihn mit dem sog. *Kalman-Gain*-Term gewichten und mit der Prädiktion fusionieren [55]. Kalman-Gain

$$\mathbf{K}_k = \mathbf{C}^{\mathrm{p}}_k \mathbf{H}^{\mathrm{T}}_k \left( \mathbf{C}^z_k + \mathbf{H}_k \mathbf{C}^{\mathrm{p}}_k \mathbf{H}^{\mathrm{T}}_k \right)^{-1} \tag{2.2}$$

gibt an, wie stark der Korrekturterm in die Schätzung eingeht: intuitiv, je größer die Messunsicherheit $\mathbf{C}^z_k$ relativ zur Prädiktionsunsicherheit, desto kleiner wird der Korrekturterm gewichtet, und umgekehrt:

$$\underline{\hat{x}}^{\mathrm{e}}_k = \underline{\hat{x}}^{\mathrm{p}}_k + \mathbf{K}_k \left( \underline{\hat{z}}_k - \mathbf{H}_k \underline{\hat{x}}^{\mathrm{p}}_k \right) \tag{2.3a}$$
$$\mathbf{C}^{\mathrm{e}}_k = \mathbf{C}^{\mathrm{p}}_k - \mathbf{K}_k \mathbf{H}_k \mathbf{C}^{\mathrm{p}}_k. \tag{2.3b}$$

Das Kalman-Filter hat die BLUE-Eigenschaft (engl. *best linear unbiased estimator*, dt. „bester linearer erwartungswertreuer Schätzer“ [60]), d. h. für lineare Systeme

2 In der Zustandsschätzung wird für die Größe des Zustandsvektors i. d. R. die Buchstabe $N$ verwendet, in der Kryptographie steht sie aber oft für die Anzahl der Parteien im Protokoll. Um Missverständnisse zu vermeiden, bezeichnen wir im Folgenden die Zustandsgröße mit $L$ und die Parteienanzahl mit $N$.

unter den getroffenen Annahmen sind seine Schätzungen nachweisbar optimal. Für die genaue Herleitung verweisen wir auf [47, 40, 55].

Für nicht-lineare Systeme existieren Varianten des Kalman-Filters, die das Systemmodell approximieren, z. B. durch die Linearisierung um den aktuellen Schätzwert (*Extended Kalman Filter*, EKF) [60, Abschn. 10.2] oder durch gezieltes Sampling der Verteilungen der Zufallsvariablen (*Unscented Kalman Filter*, UKF) [45].

### 2.1.2 Informationsfilter

Das Informationsfilter (IF) ist eine algebraische Umformulierung des Kalman-Filters, in der die Fusion mehrerer gleichzeitig aufgenommener Messungen sich effizienter durchführen lässt [61]. In der klassischen Formulierung des KF müssen Messungen entweder einzeln nacheinander in den prädizierten Zustand fusioniert werden, oder zu einem (sehr langen) Gesamtmessvektor konkateniert und in einem großen Filterschritt verarbeitet werden. Weil beide Optionen für die filternde Instanz, z. B. den zentralen Knoten in einem verteilten Sensornetzwerk, sehr rechenintensiv werden können, wird in diesem Fall oft die Informationsform verwendet.

Im IF werden statt dem Systemzustandsvektor $\underline{\hat{x}}_k$ und seiner Kovarianzmatrix $\mathbf{C}_k$ der sog. „Informationsvektor“ $\underline{\hat{y}}_k$ und die „Infomationsmatrix“ $\mathbf{Y}_k$ betrachtet:

$$\underline{\hat{y}}_k = \mathbf{C}_k^{-1} \underline{\hat{x}}_k, \tag{2.4a}$$

$$\mathbf{Y}_k = \mathbf{C}_k^{-1}. \tag{2.4b}$$

Analog werden auch die Messungen transformiert:

$$\underline{\hat{i}}_k = \mathbf{H}_k^{\mathrm{T}} \left(\mathbf{C}_k^z\right)^{-1} \underline{\hat{z}}_k, \tag{2.5a}$$

$$\mathbf{I}_k = \mathbf{H}_k^{\mathrm{T}} \left(\mathbf{C}_k^z\right)^{-1} \mathbf{H}_k. \tag{2.5b}$$

Wenn wir nun $N$ Messungen zu einem Zeitpunkt haben, lassen sich diese in der Informationsform mit einfacher Addition fusionieren und zum Filtern verwendet werden. Gegeben die Messmatrix $\mathbf{H}_k^j$, den Messvektor $\underline{\hat{z}}_k^j$, die Messkovarianzmatrix $\mathbf{C}_k^{z,j}$ und die Informationsformen $\underline{\hat{i}}_k^j, \mathbf{I}_k^j$ davon für den Sensor $j \in \{1, \ldots, N\}$, gilt

$$\underline{\hat{y}}_k^{\mathrm{e}} = \underline{\hat{y}}_k^{\mathrm{p}} + \sum_{j=1}^{N} \underline{\hat{i}}_k^j = \left(\mathbf{C}_k^{\mathrm{p}}\right)^{-1} \underline{\hat{x}}_k^{\mathrm{p}} + \sum_{j=1}^{N} \left(\mathbf{H}_k^j\right)^{\mathrm{T}} \left(\mathbf{C}_k^{z,j}\right)^{-1} \underline{\hat{z}}_k^j \quad \text{und} \tag{2.6a}$$

$$\mathbf{Y}_k^{\mathrm{e}} = \mathbf{Y}_k^{\mathrm{p}} + \sum_{j=1}^{N} \mathbf{I}_k^j = \left(\mathbf{C}_k^{\mathrm{p}}\right)^{-1} + \sum_{j=1}^{N} \left(\mathbf{H}_k^j\right)^{\mathrm{T}} \left(\mathbf{C}_k^{z,j}\right)^{-1} \mathbf{H}_k^j. \tag{2.6b}$$

Der Prädiktionsschritt in der Informationsform ist etwas komplizierter, wird aber hier nicht weiter ausgeführt, weil er im Folgenden nicht relevant sein wird. Die interessierten Leser verweisen wir auf [61, Abschn. 1.4.2].

Ein großer Vorteil des Informationsfilters bei einer verteilten Zustandsschätzung (z. B. in einem Sensornetzwerk) gegenüber dem klassischen Kalman-Filter ist, dass sich der Rechenaufwand des Filterschritts auf die einzelnen Netzwerkknoten verteilen und damit stark parallelisieren lässt, indem die zentral liegenden Knoten die Messungen ihrer Nachbarn in Informationsform sammeln und voraggregieren.

### 2.1.3 Konsensfilter

Das *Kalman-Konsensfilter* ist ein Verfahren für die verteilte Zustandsschätzung in einem Netzwerk aus Sensoren bzw. Schätzer, in dem, im Gegensatz zum Informationsfilter, kein zentraler Knoten das globale Filterergebnis berechnet, sondern alle Knoten sich gemeinsam auf eine Konsensschätzung einigen. Dies ist möglich, weil wir davon ausgehen, dass alle Knoten den gleichen Systemzustand messen bzw. schätzen [61]. Für das Verfahren existieren grundsätzlich zwei Ansätze: Konsensbildung über die einzelnen Messungen [62] und über die lokale Schätzungen [63].

In [63] wird ein Kalman-Konsensfilter beschrieben, in dem jeder Netzwerkknoten $j$ eine lokal optimale Schätzung eines externen Zustands berechnet (z. B. mittels eines lokalen Kalman-Filters) und nur mit seinen Nachbarn $\mathcal{N}_j \subset \{1, \ldots, N\}$ verbunden ist. In jedem Zeitschritt $k$ sendet jeder Knoten seine aktuelle Zustandsprädiktion $\underline{\hat{x}}_k^{j,\mathrm{p}}$ und seine aktuelle Messung in Informationsform $\underline{\hat{i}}_k^j, \mathbf{I}_k^j$ an seine Nachbarn $\mathcal{N}_j$. Anschließend berechnet jeder Knoten $j$ lokal

$$
\begin{aligned}
\underline{\hat{i}}_k^{j+} &= \underline{\hat{i}}_k^j + \sum_{n \in \mathcal{N}_j} \underline{\hat{i}}_k^n \\
\mathbf{I}_k^{j+} &= \mathbf{I}_k^j + \sum_{n \in \mathcal{N}_j} \mathbf{I}_k^n \\
\mathbf{C}_k^{j,\mathrm{e}} &= \left( \left( \mathbf{C}_k^{j,\mathrm{p}} \right)^{-1} + \mathbf{I}_k^{j+} \right)^{-1} \\
\underline{\hat{x}}_k^{j,\mathrm{e}} &= \underline{\hat{x}}_k^{j,\mathrm{p}} + \mathbf{C}_k^{j,\mathrm{e}} \left( \underline{\hat{i}}_k^{j+} - \mathbf{I}_k^{j+} \underline{\hat{x}}_k^{j,\mathrm{p}} \right) + \epsilon \cdot \mathbf{C}_k^{j,\mathrm{e}} \sum_{n \in \mathcal{N}_j} \left( \underline{\hat{x}}_k^{n,\mathrm{p}} - \underline{\hat{x}}_k^{j,\mathrm{p}} \right)
\end{aligned}
\tag{2.7}
$$

mit einem einstellbaren zeitinvarianten Updategewicht $\epsilon \in [0, 1]$. Zu beachten ist, dass die $\underline{\hat{i}}_k^{j+}, \mathbf{I}_k^{j+}$ hier nur die Nachbardaten enthalten und nicht die globale Aggregate $\underline{\hat{i}}_k, \mathbf{I}_k$ aus den Gleichungen (2.6a) und (2.6b) sind.

Das Kalman-Konsensfilter hat gegenüber anderen verteilten Filterverfahren die Vorteile, dass es keine zentrale Instanz im Netzwerk voraussetzt, wodurch die aktuelle globale Schätzung grundsätzlich jederzeit bei jedem Knoten vorliegt, und dass die Kommunikation ausschließlich unter den Nachbarknoten stattfindet. Auf der anderen Seite, weil kein Knoten *alle* aktuell vorliegende Messungen fusioniert, kann keine

Optimalität der lokalen Schätzungen garantiert werden und es kann zu Diskrepanzen zwischen den Schätzungen einzelner Knoten kommen [61].

**Gossip-Konsensfilter** Die aktuelle Forschung der sicheren Konsensfilter befasst sich allerdings mit einer einfacheren Variante namens „Gossip-Konsensfilter“ (vom engl. *gossip* – „Klatsch“) [30, 67]. In diesem Verfahren berechnet jeder Netzwerknoten seine Zustandsschätzung lokal und tauscht mit seinen jeweiligen Nachbarn solange Informationen darüber aus, bis das gesamte Netzwerk iterativ auf ein globales Mittelwert konvergiert. Der Updateschritt für die Gossip-Runde $t$ ist dann

$$\underline{\hat{x}}^{j,\mathrm{e}}_{k,t+1} = \underline{\hat{x}}^{j,\mathrm{e}}_{k,t} + \sum_{n\in\mathcal{N}_j} w^{jn}_{k,t}\left(\underline{\hat{x}}^{n,\mathrm{e}}_{k,t} - \underline{\hat{x}}^{j,\mathrm{e}}_{k,t}\right) = \underbrace{\left(1 - \sum_{n\in\mathcal{N}_j} w^{jn}_{k,t}\right)}_{w^{jj}_{k,t}} \underline{\hat{x}}^{j,\mathrm{e}}_{k,t} + \sum_{n\in\mathcal{N}_j} w^{jn}_{k,t}\underline{\hat{x}}^{n,\mathrm{e}}_{k,t}, \quad (2.8)$$

wobei $w^{nj}_{k,t} > 0$ die Gewichtungen der einzelnen Nachbarschätzungen für der Fusion sind. Dieser Updateschritt ist ähnlich dem (2.7), allerdings wird dabei nicht die Kovarianzmatrix der Schätzung aktualisiert.

O. B. d. A. befassen wir uns im Folgenden nur mit skalaren zeitinvarianten symmetrischen Gewichten, sodass $w^{xy} = w^{yx} \in (0, 1]$, und skalaren Systemzuständen $\boldsymbol{x}^j_{k,t}$. Schreiben wir die Gewichte als eine symmetrische Adjazenzmatrix $\mathbf{W}$, deren Elemente gleich $w_{jn}$, wenn $n \in \mathcal{N}_j \cup \{j\}$, und ansonsten 0 sind, konvergiert das Gossip-Verfahren asymptotisch zu einem Mittelwert aller Schätzungen gdw. $\mathbf{W}$ zeilenstochastisch ist [31]. Wir nennen $\mathbf{W}$ im Folgenden die „Gewichtungsmatrix“.

**Gossip-Filter in endlichen Körpern** Das vertrauliche Gossip-Verfahren in [30] baut auf der Arbeit einer anderen Forschungsgruppe auf, die sich mit dem Konsensfilter in endlichen Körpern befasste und einen Algorithmus mit starken Konvergenzgarantien entwickelte [65]. Sie zeigen, dass in einem endlichen Körper $\mathbb{Z}_s \equiv \mathbb{Z}/s\mathbb{Z}$ die asymptotische Konvergenz eines Gossip-Verfahrens die Konvergenz in endlicher Zeit impliziert, und dass wenn

1. die Parteianzahl $N$ und das Modulus $s$ von $\mathbb{Z}_s$ teilerfremd sind,
2. alle $\hat{x}^j_k$ kleiner oder gleich $\frac{s}{N}$ sind, d. h. $s \geq N \cdot \max_{j,k} \hat{x}^j_k$,
3. das charakteristische Polynom der Gewichtungsmatrix $\mathbf{W}_s \in \mathbb{Z}_s^{N\times N}$ gleich $s^{N-1}(s-1)$ ist, d. h. wenn ihre Eigenwerte $\sigma(\mathbf{W}_s) = \{1, 0, \ldots, 0\}$ sind, und
4. die Matrix $\mathbf{W}_s$ doppelt-stochastisch in $\mathbb{Z}_s$ ist[3],

3 Dabei ist zu beachten, dass die Elemente einer doppelt-stochastischen Matrix in $\mathbb{Z}_s$ nicht auf das Intervall $[0, 1]$ wie in $\mathbb{R}$ beschränkt sind, sondern dank der modularen Addition auf dem kompletten Definitionsbereich $\{0, \ldots, s-1\}$ verteilt sein können.

dann konvergieren alle Schätzungen zu einem globalen Mittelwert nach maximal $T$ Gossip-Runden, wobei $T$ die Dimension des größten Jordanblocks ist, der mit einem der Eigenwerte 0 der $\mathbf{W}_s$ assoziiert ist.

Weil das Berechnen einer Matrix, die die Bedingungen 3 und 4 erfüllt, im allgemeinen $\mathcal{NP}$-hart ist, zeigen die Autoren in [65] stattdessen, wie große Gewichtungsmatrizen effizient aus kleineren (die die notwendigen Bedingungen erfüllen) mittels Kronecker-Produkts erzeugt werden können. Diese Methode setzt aber voraus, dass das Netzwerk nach der vorberechneten Adjazenzmatrix aufgebaut wird, und bietet keine offensichtliche Möglichkeit an, diese Matrix für ein bestehendes Netzwerk effizient zu berechnen bzw. sie im Betrieb um zusätzliche Sensoren zu erweitern.

Das vertrauliche Gossip-Verfahren in [30] nimmt an, dass die Matrix $\mathbf{W}_s$ bereits offline vorberechnet und ihre Zeilen an die entsprechenden Parteien verteilt wurden. Die Autoren zeigen, dass der Updateschritt (2.8) auch homomorph im Paillier-Schema (s. Def. 2.6) berechnet werden kann, obwohl das Klartextmodulus $s$ darin keine Primzahl ist und $\mathbb{Z}_s$ dadurch im Allgemeinen kein Körper ist, sowie dass die Bedingungen 1 und 2 unter sinnvollen Annahmen über die Schlüssellänge mit überwältigender Wahrscheinlichkeit erfüllt sind. Die Autoren schlagen aber kein konkretes Gossip-Protokoll vor (v. a. ist unklar, wer und wie die Filterergebnisse entschlüsseln und verwenden soll), und die von ihnen vorgeschlagene „sichere Cloud-Architektur“ negiert den Vorteil der lokalen Kommunikation, den ein Gossip-Filter mit sich bringt.

**Gossip-Filter mit Gewichtsmaskierung** Der andere Vorschlag [67] hat dagegen eine sehr klare Protokolldefinition, die direkt auf der Updateregel (2.8) aufbaut. Das Protokoll sieht vor, dass in jedem benachbarten Knotenpaar $P_j, P_n$ jede Partei ein eigenes Paillier-Schlüsselpaar besitzt, ihre jeweiligen Schätzung $\hat{x}^j_{k,t}, \hat{x}^n_{k,t}$ damit verschlüsselt, und sich gegenseitig sendet. Jede Partei bildet für die andere homomorph die Differenz $\Delta\hat{x}^{jn}_{k,t} = \hat{x}^n_{k,t} - \hat{x}^j_{k,t}$ und multipliziert sie ebenfalls homomorph mit einem (quantisierten) Gewicht $w_j, w_n$, den sie vorher gleichverteilt aus dem Intervall $[0, \bar{w}]$ mit einer global einstellbaren $\bar{w} \in (0, 1)$ zieht. Diese Differenzen werden dann für $T$ Runden von allen Nachbarn gesammelt und in die eigene Schätzung fusioniert.

Die Autoren zeigen, dass derart fusionierte Schätzungen zu einem globalen Mittelwert $\alpha = \frac{1}{N}\sum_{j=1}^{N} \hat{x}^j_{k,1}$ konvergieren. Allerdings sind die von ihnen diskutierten Sicherheitsgarantien etwas zweifelhaft, denn sie behaupten, eine Partei $P_j$ könnte den Zustand $\hat{x}^n_{k,t}$ ihres Nachbars $P_n$ aus $w_n \cdot \Delta\hat{x}^{jn}_{k,t}$ nicht berechnen, weil $w_n$ von $P_n$ zufällig gewählt wurde. Weil $w_n$ allerdings auf das reelle Intervall $[0, 1)$ beschränkt ist, lernt $P_j$ zwar nicht den genauen Wert von $\hat{x}^n_{k,t}$, aber ob $\hat{x}^n_{k,t}$ größer oder kleiner als $\hat{x}^j_{k,t}$ ist, denn das Vorzeichen von $\Delta\hat{x}^{jn}_{k,t}$ bleibt nach der Multiplikation mit $w_n$ unverändert (außer im Fall $w_n = 0$). Damit ist es offensichtlich, dass $P_j$ aus der

Ausführung des Protokolls partielle Informationen über die Eingabe von $P_n$ lernt und umgekehrt, was die Vertraulichkeit des Verfahrens in Frage stellen lässt.

## 2.2 Kryptographie

Moderne Kryptographie ist „die Forschung der mathematischen Verfahren zur Sicherung der digitalen Informationen, Systeme und verteilten Berechnungen gegen feindliche Angriffe“ [48, Abschn. 1.1]. Ihre Hauptprinzipien sind formale Definitionen, präzise Annahmen und Sicherheitsbeweise: moderne kryptographische Verfahren erreichen mathematisch *beweisbare Sicherheit*, indem sie klare Sicherheitsdefinitionen unter bestimmten Annahmen erfüllen. Ein weiterer Grundsatz ist *Kerkhoffs' Prinzip*, das verlangt, dass kryptographische Verfahren sicher bleiben, selbst wenn ihre Funktionsweisen (aber nicht die Schlüssel) allgemein bekannt sind.

### 2.2.1 Asymptotische Sicherheit

Die Sicherheit vieler kryptographischer Verfahren wird auf gut studierte komplexitätstheoretische Annahmen zurückgeführt. Es ist zwar möglich, eine bedingungslose informationstheoretische (perfekte) Sicherheit zu erreichen, sie ist aber für die meisten realen Applikationen unpraktikabel [48, Kap. 2]. Stattdessen wird für die Sicherheitsbeweise oft angenommen, dass der Angreifer „effizient“ ist (s. unten).

**Sicherheitsparameter** Der Sicherheitsparameter $n \in \mathbb{N}$ ist ein abstrakter Parameter der kryptographischen Verfahren, sowie aller daran beteiligten Parteien (egal ob ehrlich oder vom Angreifer gesteuert), der ihre Laufzeit, aber auch die Erfolgswahrscheinlichkeit des Angreifers begrenzt. Eine Intuition für den Sicherheitsparameter könnte z. B. die Schlüssellänge in Verschlüsselungsschemas sein, sie sind aber nicht miteinander gleichzusetzen.

Der Sicherheitsparameter wird üblicherweise unär kodiert, d. h. als eine Bitfolge aus $n$ Einsen ($1^n$), damit die Laufzeit kryptographischer Algorithmen formal als „polynomiell in der *Länge* der Eingabe“ definiert werden kann. Für die Praxis hat diese Schreibweise keine weiteren Implikationen.

**Angreifer** In der formalen Kryptographie modellieren wir Angreifer (sowie Protokollpartien) als Turing-Maschinen und bezeichnen sie als „effizient“, wenn sie in einer probabilistischen polynomiellen Zeit (engl. *probabilistic polynomial time*, kurz: PPT) im Sicherheitsparameter $n$ laufen [48, Abschn. 3.1]. Dabei betrachten wir die

Maschinen selbst als deterministisch, aber mit jeweils eigenem „Zufallsband“ (engl. *random tape*) mit unabhängig und gleichverteilt gezogenen Bits ausgestattet, von dem sie ihren Zufall beziehen. Dadurch können wir Angriffe modellieren, wo der Angreifer den „Zufall“ der korrumpierten Parteien ausliest oder sogar manipuliert.

**Vernachlässigbarkeit** Wir bezeichnen eine Funktion $f : \mathbb{N} \to \mathbb{R}_{\geq 0}$ formal als vernachlässigbar (engl. *negligible*), wenn für jedes positive Polynom $p$ ein Wert $N$ existiert, sodass für jede ganze Zahl $n > N$ gilt $f(n) < \frac{1}{p(n)}$ [48, Def. 3.4]. Die Wahrscheinlichkeit P eines Ereignisses $E(n)$ ist „vernachlässigbar in $n$“, wenn eine vernachlässigbare Funktion $\mu(n)$ existiert, sodass $\mathrm{P}[E(n)] \leq \mu(n)$. Umgekehrt ist die Wahrscheinlichkeit $\mathrm{P}[E(n)]$ „überwältigend in $n$“, wenn $1 - \mathrm{P}[E(n)] \leq \mu(n)$.

Wir sagen, dass ein kryptographisches Verfahren *asymptotisch sicher* ist, wenn jeder PPT-Angreifer es nur mit einer höchstens im Sicherheitsparameter $n$ vernachlässigbaren Wahrscheinlichkeit brechen kann (wobei „brechen“ noch zu definieren ist). Intuitiv existiert für ein asymptotisch sicheres Verfahren immer ein Sicherheitsparameterwert $n$, ab dem kein Angriff darauf mehr praktikabel ist.

**Ununterscheidbarkeit** Um das „Brechen“ eines asymptotisch sicheren Verfahrens zu formalisieren, benötigen wir das Konzept der komplexitätstheoretischen Ununterscheidbarkeit (engl. *computational indistinguishability*) [52, Abschn. 6.2].

**Definition 2.1 (Komplexitätstheoretische Ununterscheidbarkeit)** Ein Wahrscheinlichkeitsensemble (engl. *probability ensemble*) $X = \{X(a,n)\}_{a\in\{0,1\}^*;n\in\mathbb{N}}$ ist eine unendliche Sequenz von Zufallsvariablen, indiziert mit der Eingabe $a$ und dem Sicherheitsparameter $n$. Zwei Wahrscheinlichkeitsensembles $X = \{X(a,n)\}_{a,n}, Y = \{Y(a,n)\}_{a,n}$ gelten als *komplexitätstheoretisch ununterscheidbar*, wenn für jeden PPT-Algorithmus $D$ („Unterscheider“) eine vernachlässigbare Funktion $\mu(n)$ existiert, sodass für jede $a \in \{0,1\}^*$ und jede $n \in \mathbb{N}$ gilt:

$$|\mathrm{P}[D(X(a,n)) = 1] - \mathrm{P}[D(Y(a,n)) = 1]| \leq \mu(n).$$

Wir schreiben dann auch $X \overset{\mathrm{c}}{\equiv} Y$. Intuitiv bedeutet diese Formulierung, dass jeder effiziente Algorithmus die beiden Verteilungen nur mit einer höchstens vernachlässigbaren Wahrscheinlichkeit voneinander unterscheiden kann.

### 2.2.2 Homomorphe Verschlüsselung

Einer der Grundbausteine der modernen Kryptographie sind die Verschlüsselungsschemas. Für diese Arbeit sind insbesondere die Schemas mit dem sog. „Chiffrat-Homomorphismus“ interessant, auch als „homomorphe Verschlüsselung“ bezeichnet.

Ein Verschlüsselungsschema wird durch die Algorithmen KeyGen (Schlüsselerzeugung), Enc (Verschlüsselung) und Dec (Entschlüsselung), sowie durch seine Klartextdomäne $\mathcal{M}$ definiert (formalisiert in der Def. 2.3). Weil KeyGen, Enc und Dec im Allgemeinen probabilistisch sind, benutzen wir im Folgenden für die Zuweisung ihrer Ausgaben das Pfeilsymbol, z. B. $sk \leftarrow \mathsf{KeyGen}(1^n)$.

**Symmetrische und asymmetrische Schemas** Verschlüsselungsschemas unterteilen sich in *symmetrische* und *asymmetrische*: in den ersteren gibt KeyGen nur einen Schlüssel $sk$ aus, der sowohl für die Verschlüsselung, als auch für die Entschlüsselung verwendet wird. In den letzteren gibt KeyGen ein Schlüsselpaar $(pk, sk)$ aus: Enc wird mit dem „öffentlichen Schlüssel" $pk$ und Dec, mit dem „Geheimschlüssel" $sk$ parametrisiert. In der vorliegenden Arbeit befassen wir uns ausschließlich mit asymmetrischen homomorphen Schemas, denn, obwohl die symmetrische homomorphe Verschlüsselung möglich ist [34], sind die meisten praktischen Umsetzungen asymmetrisch und wir benötigen die Asymmetrie für unsere Verfahren.

**Ununterscheidbarkeit der Chiffrate** Die Sicherheit eines Verschlüsselungsverfahrens wird in der Kryptographie formal über die Ununterscheidbarkeit der Chiffrate definiert. Dazu stellen wir das sog. „IND-CPA-Spiel" (engl. *INDistinguishability under Chosen-Plaintext-Attack* auf:

1. Wir berechnen $(pk, sk) \leftarrow \mathsf{KeyGen}(1^n)$ und übergeben $pk$ an den Angreifer $\mathcal{A}$.
2. $\mathcal{A}$ gibt zwei Nachrichten $m_0, m_1 \in \mathcal{M}$ der gleichen Bitlänge aus.
3. Wir ziehen $b \leftarrow \{0, 1\}$, berechnen $c \leftarrow \mathsf{Enc}(pk, m_b)$ und übergeben $c$ an $\mathcal{A}$.
4. $\mathcal{A}$ gibt $b'$ aus und gewinnt das Spiel gdw. $b = b'$.

**Definition 2.2 (IND-CPA-Sicherheit)** Ein asymmetrisches Verschlüsselungsschema $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ gilt als „IND-CPA-sicher", wenn für jeden PPT-Angreifer $\mathcal{A}$ eine vernachlässigbare Funktion $\mu(n)$ existiert, sodass $\mathcal{A}$ das beschriebene Spiel mit einer Wahrscheinlichkeit $\mathrm{P} \leq \frac{1}{2} + \mu(n)$ gewinnt. (Zitiert nach [48, Def. 11.2].)

Intuitiv, wenn kein Angreifer einen signifikanten Vorteil ggü. Raten hat, obwohl er $m_0, m_1$ selbst aussuchen durfte, verrät das Chiffrat ihm nichts über seinen Inhalt.

Im IND-CCA-Spiel (engl. *Chosen-Ciphertext-Attack*) wird der Angreifer verstärkt, indem wir ihm zusätzlich den Zugriff auf eine Orakelfunktionalität geben, die ihm beliebige Chiffrate (außer $c$) entschlüsselt. Dabei unterscheiden wir zwischen CCA1-Spiel, in dem er das Orakel ab dem Schritt 3 nicht mehr verwenden kann, und CCA2, in dem das Orakel durchgehend verfügbar ist. Obwohl IND-CCA2-sichere

| Operation | Abkürzung | Bedeutung |
|---|---|---|
| $\mathsf{Enc}(pk, m)$ | $[\![m]\!]_{pk}$ bzw. $[\![m]\!]$ | Verschlüsselung |
| $\mathsf{Eval}(pk, +, c_1, c_2)$ | $c_1 +_{pk} c_2$ | Homomorphe Addition |
| $\mathsf{Eval}(pk, -, c_1, c_2)$ | $c_1 -_{pk} c_2$ | Homomorphe Subtraktion |
| $\mathsf{Eval}(pk, \times, c, m)$ | $c \odot_{pk} m$ | Homomorphe Multiplikation (s. unten) |

**Tabelle 2.1:** Im Folgenden werden die Schreibweisen der häufigsten homomorphen Operationen zwecks Übersichtlichkeit abgekürzt. $\odot_{pk}$ bezeichnet die homomorphe Multiplikation eines Chiffrats $c$ mit einem Klartext-Integer $m$ in additiv homomorphen Schemas.

Verschlüsselungsschemas existieren (z. B. Cramer–Shoup [18]), kann kein solches Schema homomorph sein und umgekehrt [9].

**Chiffrat-Homomorphismus** In einem homomorphen Verschlüsselungsschema sind die Chiffrate verformbar (engl. *malleable*), sodass wir mathematische Operationen auf verschlüsselten Nachrichten ausführen können, ohne sie entschlüsseln zu müssen. Wir formalisieren dies, indem wir die drei Algorithmen (KeyGen, Enc, Dec) um einen vierten, Eval (Evaluieren), ergänzen.

**Definition 2.3** Ein homomorphes Verschlüsselungsschema (KeyGen, Enc, Dec, Eval) besteht aus vier Prozeduren:

- $(pk, sk) \leftarrow \mathsf{KeyGen}(1^n)$ nimmt den Sicherheitsparameter und gibt ein Schlüsselpaar aus.[4]
- $c \leftarrow \mathsf{Enc}(pk, b)$ nimmt den öffentlichen Schlüssel und ein Klartextbit und gibt ein Chiffrat dazu aus.
- $b \leftarrow \mathsf{Dec}(sk, c)$ nimmt den Geheimschlüssel und ein Chiffrat und gibt ein Klartextbit aus.
- $\underline{c}' \leftarrow \mathsf{Eval}(pk, \Pi, \underline{c})$ nimmt den öffentlichen Schlüssel, einen logischen Schaltkreis $\Pi$ und einen Vektor aus Chiffraten $\underline{c} = (c_1, \ldots, c_t)$ (eins für jedes Eingabebit von $\Pi$) und gibt einen Vektor $\underline{c}'$ der Chiffrate (eins für jedes Ausgabebit) aus.

(Zitiert nach [39, Def. 5.2.1])

Wir bezeichnen die Ausgaben von Enc im Folgenden als „frische Chiffrate“ und die von Eval als „evaluierte Chiffrate“.

4 In [39] wird außerdem ein „Funktionalitätsparameter“ $\tau$ für KeyGen definiert, der in den sog. „etwas homomorphen Schemas“ begrenzt, wie oft ein frisches Chiffrat homomorph evaluiert werden kann, bevor es sich nicht mehr korrekt entschlüsseln lässt. Weil in dieser Arbeit aber kein solches Schema vorkommt, lassen wir $\tau$ aus unseren Definitionen aus.

**Definition 2.4 (Perfekte Korrektheit)** Bezeichnen wir mit (KeyGen, Enc, Dec, Eval) ein homomorphes Verschlüsselungsschema und mit $\mathcal{C}$ eine Schaltkreisfamilie. Wir sagen, dass dieses Schema *perfekt korrekt* für $\mathcal{C}$ ist, wenn es sowohl frische, als auch evaluierte Chiffrate korrekt entschlüsselt, d. h. für jeden $n \in \mathbb{N}$ müssen zwei Bedingungen gelten:

- Für jedes $b \in \{0, 1\}$:

$$\mathrm{P}\left[\mathsf{Dec}(sk, c) = b : (pk, sk) \leftarrow \mathsf{KeyGen}(1^n), c \leftarrow \mathsf{Enc}(pk, b)\right] = 1;$$

- Für jeden $\Pi \in \mathcal{C}$ und jede Bitfolge $\underline{b} = (b_1, \ldots, b_t) \in \{0, 1\}^t$ (eins für jedes Eingabebit von $\Pi$):

$$\mathrm{P}\left[\mathsf{Dec}(sk, \underline{c}') = \Pi(\underline{b}) : \begin{array}{c}(pk, sk) \leftarrow \mathsf{KeyGen}(1^n), \\ \underline{c} \leftarrow \mathsf{Enc}(pk, \underline{b}), \underline{c}' \leftarrow \mathsf{Eval}(pk, \Pi, \underline{c})\end{array}\right] = 1.$$

Die Wahrscheinlichkeiten sind hier formal über den verwendeten Zufall definiert. (Zitiert nach [39, Def. 5.2.2].)

Die allgemeine (nicht perfekte) Korrektheit wird dadurch definiert, dass die Chiffrate nur *mit einer überwältigender Wahrscheinlichkeit* korrekt entschlüsselt werden (statt mit $\mathrm{P} = 1$), die Def. 2.4 wird aber von vielen praktischen Schemas erfüllt und vereinfacht unsere spätere Sicherheitsbeweise erheblich. Im Folgenden sprechen wir bei den Eingaben für Enc und Eval oft nicht mehr von Bits und Bitfolgen, sondern von Integers bzw. Integer-Arrays (Vektoren und Matrizen). Integer-Arrays werden dabei elementweise unter Erhaltung deren Struktur verschlüsselt und evaluiert.

Weil die IND-CPA-Sicherheit unabhängig von der Eval-Funktionalität definiert war, wird sie davon nicht beeinträchtigt [34, 39]. Der Homomorphismus schließt aber die IND-CCA2-Sicherheit aus, die mit der *Un*verformbakeit der Chiffrate äquivalent ist [9]. Dass die IND-CPA-Sicherheit bei der Verschlüsselung von Klartext-Arrays gleicher Länge erhalten bleibt, folgt direkt aus dem [48, Satz 11.6].

**Klassifikation** Abhängig von der berechenbaren Schaltkreismenge $\mathcal{C}$ können homomorphe Schemas weiter unterteilt werden [39]:

- **Teilhomomorphe Verschlüsselung** (engl. *partially homomorphic encryption*) ermöglicht entweder die homomorphe Addition oder die homomorphe Multiplikation der Chiffrate, aber nicht beides. *Additiv homomorphe* Schemas sind z. B. die von Goldwasser–Micali [37] und von Paillier [64], während die von Rivest–Shamir–Adleman (RSA) [46] und von ElGamal [22] zu den *multiplikativ homomorphen* gehören.

- **Etwas homomorphe Verschlüsselung** (engl. *leveled/somewhat homomorphic encryption*) ermöglicht homomorphe Auswertung beliebiger Schaltkreise bis zu einer einstellbaren Tiefe und Komplexität. Intuitiv sind darin sowohl die Addition, als auch die Multiplikation homomorph berechenbar, aber die Korrektheit ist nur bis zu einer maximalen Anzahl der Evaluationen garantiert.
- **Vollhomomorphe Verschlüsselung** (engl. *fully homomorphic encryption*) ermöglicht die homomorphe Auswertung beliebiger Schaltkreise, d. h. Addition und Multiplikation, ohne weiteren Einschränkungen. Später in diesem Abschnitt gehen wir auf diese Kategorie nochmal näher ein.

Neben der IND-CPA-Sicherheit können alle homomorphe Verschlüsselungen eine Reihe weiterer Eigenschaften haben, z. B. Kompaktheit, Funktionsvertraulichkeit und Multihop-Fähigkeit [39, Abschn. 5.2.2], wir benötigen für die Beweise später aber nur eine, die alle anderen zusammenfasst.

**Definition 2.5 (Komplexitätstheoretisch starker Homomorphismus)** Wir sagen, dass ein Schema $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ *komplexitätstheoretisch stark homomorph* für eine Schaltkreisfamilie $\mathcal{C}$ ist, wenn für jedes $\Pi \in \mathcal{C}$ und jede Klartext-Bitfolge $\underline{b} = (b_1, \ldots, b_t) \in \{0,1\}^t$ (eins für jedes Eingabebit von $\Pi$), folgende Wahrscheinlichkeitsensembles komplexitätstheoretisch ununterscheidbar sind:

$$\mathsf{Fresh}^*_{\Pi,\underline{b}} \stackrel{def}{=} \left\{ (r, \underline{c}, \underline{c}') : \begin{array}{c} r \leftarrow \$, (pk, sk) := \mathsf{KeyGen}(1^n; r), \\ \underline{c} \leftarrow \mathsf{Enc}(pk, \underline{b}), \underline{c}' \leftarrow \mathsf{Enc}(pk, \Pi(\underline{b})) \end{array} \right\}$$

$$\mathsf{Eval}^*_{\Pi,\underline{b}} \stackrel{def}{=} \left\{ (r, \underline{c}, \underline{c}') : \begin{array}{c} r \leftarrow \$, (pk, sk) := \mathsf{KeyGen}(1^n; r), \\ \underline{c} \leftarrow \mathsf{Enc}(pk, \underline{b}), \underline{c}' \leftarrow \mathsf{Eval}(pk, \Pi, \underline{c}) \end{array} \right\},$$

wobei wir mit $r \leftarrow \$$ das Erzeugen einer Folge $r$ aus unabhängig und gleichverteilt gezogenen Bits bezeichnen. (Zitiert nach [39, Def. 5.2.6].)

Intuitiv bedeutet die Def. 2.5, dass kein PPT-Angreifer an einem Chiffrat erkennen kann, ob es frisch ist oder evaluiert wurde, sogar wenn er den Zufall kennt, aus dem der Geheimschlüssel erzeugt wurde, und ihn damit duplizieren könnte.

**Paillier-Schema** Unser Hauptwerkzeug im Folgenden ist das additiv homomorphe asymmetrische Verschlüsselungsschema von Pascal Paillier [64].

**Definition 2.6 (Paillier-Verschlüsselungschema)** Wir definieren das asymmetrische Verschlüsselungsschema $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ wie folgt:

- **Schlüsselerzeugung** $(pk, sk) \leftarrow \mathsf{KeyGen}(1^n)$:

1. Ziehe zwei Primzahlen $p, q$ der gleichen Bitlänge[5] und berechne $s = pq$ und $\nu = \mathrm{lcm}(p-1, q-1)$, wobei lcm für das kleinste gemeinsame Vielfache steht.
2. Ziehe $g \leftarrow \mathbb{Z}_{s^2}$, sodass $\gcd(L_n(g^\nu \bmod s^2), s) = 1$ gilt, wobei gcd für den größten gemeinsamen Teiler steht, $L_s(x) \equiv \left\lfloor \frac{x-1}{s} \right\rfloor$, und $\lfloor \cdot \rfloor$ die Abrundung zur nächstkleineren ganzen Zahl repräsentiert.
3. Gebe die Tupel $pk = (s, g), sk = (p, q)$ aus. Der Nachrichtenraum $\mathcal{M}$ ist $\mathbb{Z}_s \equiv \mathbb{Z}/s\mathbb{Z}$.

- **Verschlüsselung** $c \leftarrow \mathsf{Enc}(pk, m)$ mit $pk = (s, g), m \in \mathcal{M}$:
  1. Ziehe $r \leftarrow \mathbb{Z}_s$.
  2. Berechne $c \coloneqq g^m \cdot r^s \mod s^2$ und gebe $c$ aus.
- **Entschlüsselung** $m \coloneqq \mathsf{Dec}(sk, c)$ mit $sk = (p, q), c \in \mathbb{Z}_{s^2}$: Berechne
$$m \coloneqq \frac{L_s(c^\nu \bmod s^2)}{L_s(g^\nu \bmod s^2)} \mod s$$
und gebe $m$ aus. (Zitiert nach [64].)

Den Wert $s = pq \in \mathbb{N}$ bezeichnen wir im Folgenden mit „Klartextmodulus“, weil er den Klartextraum $\mathcal{M} = \mathbb{Z}_s$ definiert. Die Eigenschaften der perfekten Korrektheit und der IND-CPA-Sicherheit des Paillier-Schemas wurden bereits in dem Originalpaper gezeigt, ebenso wie die homomorphe Eigenschaften seiner Chiffrate: für $m_1, m_2 \in \mathcal{M}$ gilt darin nämlich

$$\begin{aligned}
[\![m_1]\!] +_{pk} [\![m_2]\!] &\equiv [\![m_1]\!] \cdot [\![m_2]\!] = g^{m_1+m_2} \cdot (r_1 r_2)^s \pmod{s^2} \\
[\![m_1]\!] -_{pk} [\![m_2]\!] &\equiv [\![m_1]\!] \cdot [\![m_2]\!]^{-1} \pmod{s^2} \\
[\![m_1]\!] \odot_{pk} m_2 &\equiv [\![m_1]\!]^{m_2} \pmod{s^2}.
\end{aligned}$$

Intuitiv entspricht die homomorphe Addition der Chiffrate ihrer Multiplikation modulo $s^2$. Dies ermöglicht die homomorphe Multiplikation mit Klartext-Integer (durch $m_2$-fache homomorphe Addition des Chiffrats mit sich selbst) und homomorphe Subtraktion (durch Multiplikation mit der modularen Inverse des Subtrahend-Chiffrats).

Es ist aber nicht schwer zu erkennen, dass das Schema nicht stark homomorph nach der Def. 2.5 sein kann, weil alle homomorphe Operationen darin deterministisch sind. Es ist deshalb trivial, gegeben zwei Chiffrate, ihre homomorphe Summe von einem frischen Chiffrat der Summe ihrer Klartexte zu unterscheiden. Wenn Eval allerdings zu jeder seinen Ausgabe ein frisches Chiffrat von Null homomorph addiert, beeinträchtigt es nicht die Korrektheit, macht aber die Eval-Ausgaben von den Enc-Ausgaben ununterscheidbar und damit das gesamte Schema komplexitätstheoretisch stark homomorph. Im Folgenden beziehen wir uns ausschließlich auf diese Variante.

5 Mehr zur konkreten Schlüssellänge und wie sie mit $n$ zusammenhängt s. auf Seite 18.

**Vollhomomorphe Verschlüsselung** Das erste voll funktionstüchtige vollhomomorphe Verschlüsselungsschema wurde 2009 von Craig Gentry beschrieben [32, 33]. Es ergänzte ein etwas homomorphes Basisschema um die sog. „Bootstrapping"-Prozedur, die aus einem evaluierten Chiffrat effektiv ein frisches mit dem gleichen Klartext erzeugt, indem sie den eigenen Dec-Schaltkreis homomorph darauf ausführt. Das Originalschema von Gentry war extrem rechenintensiv (z. B. dauerte die Addition zweier 32-bit Zahlen darin über 900 s [76]), sein Durchbruch wurde aber schnell von vielen weiteren Konstruktionen gefolgt, die immer effizienter wurden.

Weil allerdings selbst die neuesten vollhomomorphen Verschlüsselungsschemas zu rechenaufwendig bzw. zu restriktiv für die zeitnahe Verarbeitung großer Datenmengen sind [19, 26], befassen wir uns in dieser Arbeit mit viel einfacheren, aber besser abschätzbaren additiv homomorphen Verfahren, v. a. dem Paillier-Schema. Für eine umfassende Übersicht der aktuellen Entwicklungen auf dem Gebiet der vollhomomorphen Verschlüsselung verweisen wir die interessierten Leser auf [1, 39].

**Praktische Aspekte** Der Sicherheitsparameter $n$ bestimmt grundsätzlich die Länge des Schlüssels, den die ehrlichen Parteien benutzen: ein größerer Wert $n$ entspricht einem längeren Schlüssel und somit einer kleineren Erfolgswahrscheinlichkeit des Angreifers [48, Abschn. 3.1.1]. Die genaue Relation zwischen diesen drei Werten hängt allerdings stark von der Konstruktion des konkreten Schemas ab. Wird die Schlüssellänge für eine konkrete Applikation gesucht, ist es deshalb oft zielführender, statt der asymptotischen Sicherheit und dem Sicherheitsparameter von der „konkreten Sicherheit" (engl. *concrete security*) und dem „Sicherheitslevel" (engl. *security level*) bzw. der „Bit-Sicherheit" (engl. *bit security*) zu reden [50, 58].

Eine „$\eta$-Bit-Sicherheit" bedeutet intuitiv, dass jeder *erfolgreiche* Angriff auf das Schema mit einem Aufwand von mindestens $O(2^\eta)$ verbunden ist, bzw. äquivalent dass jeder *effiziente* Angriff eine Erfolgswahrscheinlichkeit $\epsilon \leq 2^{-\eta}$ hat [58]. Für symmetrische Schemas, z. B. AES (engl. *Advanced Encryption Standard*) [28], sind der Sicherheitslevel und die Bitlänge des Schlüssels i. d. R. gleich, weil kein Angriff darauf bekannt ist, der effizienter als die vollständige Suche des kompletten Schlüsselraums wäre [50]. In asymmetrischen Schemas wie RSA ist die Schlüssellänge ein Vielfaches des Sicherheitslevels, weil der Angreifer die mathematischen Eigenschaften des Schlüssels ausnutzen kann, um die Suche danach effizienter zu gestalten. Ein 3072 Bit RSA-Schlüssel bietet z. B. nur eine 128-Bit-Sicherheit, d. h. die gleiche konkrete Sicherheitsgarantien wie ein 128 Bit AES-Schlüssel [5].

Wegen der ständig steigenden Rechenkapazität der modernen Technik steigen andauernd auch die Sicherheitsansprüche an die Verschlüsselung. Dies bedeutet, dass die minimale Schlüssellänge für alle Verfahren mit der Zeit stetig wächst [50]. Für das

Paillier-Schema gibt es in der Literatur diesbezüglich keine konkreten Empfehlungen, es gehört aber zusammen mit RSA zu der Kategorie der asymmetrischen Schemas, die auf Integer-Faktorisierung aufbauen [5, 50].[6] Das Bundesamt für Sicherheit in der Informationstechnik empfiehlt für die RSA-OEAP-Variante bis Ende 2022 die Klartextmoduluslänge von 2000 Bit [14, Kapitel 3.5], deshalb orientieren wir uns im Folgenden an die gleiche Empfehlung für Paillier, auch wenn die Annahme, dass für zwei Verfahren aus derselben Kategorie die gleichen Schlüssellängenempfehlungen gelten, im Allgemeinen nicht gültig ist.

Im Schritt 2 des IND-CPA-Spiels wird außerdem gefordert, dass $|m_1| = |m_2|$. Dies ist notwendig, weil die Klartextlänge sich im Allgemeinen durch die Verschlüsselung nicht verstecken lässt, und impliziert, dass die Ununterscheidbarkeit im Allgemeinen nur für Chiffrate der Klartexte gleicher Länge gilt. Weil unsere Nachrichten im Folgenden grundsätzlich Zahlen sind und um unsere Beweise später zu vereinfachen, definieren wir, dass im Folgenden alle Klartexte vor dem Verschlüsseln mit führenden Nullen auf die maximale zulässige Klartextbitlänge erweitert („gepaddet“) werden. Damit wird sichergestellt, dass alle unter dem gleichen Schlüssel verschlüsselte Klartexte die gleiche Bitlänge haben, nämlich die Länge des Klartextmodulus $s$ (2000 Bit), im Spezialfall des Paillier-Schemas. Wir beobachten aber, dass weil Paillier-Chiffrate modulo $s^2$ berechnet werden, jedes Chiffrat bei uns etwa 4000 Bit bzw. 500 Byte lang ist, unabhängig von der Länge des verschlüsselten Klartexts.

### 2.2.3 Sichere Mehrparteienberechnung

Wenn wir den Ansatz der beweisbaren Sicherheit von den grundlegenden Verschlüsselungsverfahren auf allgemeine verteilte Berechnungen erweitern wollen, finden wir uns auf dem Gebiet der sicheren Mehrparteienberechnung (engl. *secure multi-party computation*, kurz: SMC). Bei einer verteilten Berechnung kommunizieren mehrere individuelle Parteien miteinander, um gemeinsam eine Funktion auf ihren jeweiligen Eingaben zu berechnen. Der Zweck von SMC ist es, diese Berechnung sicher zu gestalten. Dabei gehen wir davon aus, dass ein Angreifer eine oder mehrere Parteien unter seine Kontrolle bringen („korrumpieren“) kann, mit dem Ziel, die Geheimnisse der ehrlichen Parteien herauszufinden oder die Ergebnisse der Berechnung zu verfälschen. In der SMC werden i. d. R. keine Horcher modelliert, weil die Kommunikation über als vertraulich und authentifiziert angenommene Kanäle abläuft [41].

6 Die andere große Kategorie der asymmetrischen Schemas baut auf den diskreten Logarithmen in endlichen Körpern auf [5], z. B. das kryptographische System von Taher ElGamal [22].

**Sicherheitseigenschaften** Eine Möglichkeit, uns einer formalen Sicherheitsdefinition für SMC zu nähern, ist über die Eigenschaften, die ein Protokoll besitzen muss, um intuitiv als sicher zu gelten. Die wichtigsten davon sind nach [41, Abschn. 1.1]:

- **Vertraulichkeit** (engl. *privacy*). Keine Partei soll mehr aus der Protokollausführung lernen, als ihr per Protokolldefinition zusteht, vor allem nichts über die Eingaben der anderen Parteien, was sie nicht aus ihrer eigenen Ein- und Ausgabe berechnen könnte.
- **Korrektheit** (engl. *correctness*). Jede Partei kann davon ausgehen, dass die Ausgabe, die sie erhalten hat, korrekt ist.
- **Eingabeunabhängigkeit** (engl. *independence of inputs*). Korrumpierte Parteien müssen ihre Eingaben unabhängig von denen der ehrlichen auswählen.
- **Ausgabegarantie** (engl. *guaranteed output delivery*). Korrumpierte Parteien können nicht verhindern, dass die ehrlichen ihre Ausgaben erhalten.
- **Fairness** (engl. *fairness*). Korrumpierte Parteien sollen ihre Ausgaben genau dann erhalten, wenn die ehrlichen ihre bekommen.

Diese Punkte könnten wir einzeln als Sicherheitsspiele (analog IND-CPA) formalisieren und beweisen, aber sie bilden noch keine Sicherheitsdefinition, weil wir ihre Vollständigkeit nicht garantieren können, und eine solche Definition ausreichend einfach sein muss, um offensichtlich *alle mögliche* Angriffe auszuschließen.

**Simulationsparadigma** Der Standardansatz zur Formalisierung der SMC-Sicherheit ist das sog. „Ideal/Real-Simulationsparadigma“ (engl. *ideal/real simulation paradigm*) [41]. O. B. d. A. betrachten wir den Spezialfall der Zweiparteienberechnung und stellen uns eine ideale Welt vor, in der es eine vertrauenswürdige und unkorrumpierbare dritte Partei gibt, die eine Funktionalität[7] $f$ berechnen kann, und die mit den Parteien $P_1$ und $P_2$ über perfekt vertrauliche Kanäle kommuniziert. Um $f$ in dieser Welt zu berechnen, müssen die Parteien nur ihre jeweiligen Eingaben an die vertrauenswürdige dritte Partei senden und ihre Ausgaben von ihr empfangen.

Diese Berechnung ist dann per Definition sicher, aber in der realen Welt existieren keine vertrauenswürdigen dritten Parteien. Deshalb müssen $P_1, P_2$ stattdessen ein Protokoll $\pi$ erfinden, das die ideale Funktionalität realisiert. Wir sagen, dass $\pi$ *sicher* ist, wenn kein Angreifer mehr Schaden in der realen Welt anrichten kann, als ein Angreifer in der idealen könnte, denn in der letzteren können per Definition keine Angriffe erfolgreich sein [41]. Im Folgenden formalisieren wir diese Vorstellung.

7 Weil $f$ grundsätzlich probabilistisch sein kann, wird in der Kryptographie der Begriff „Funktionalität“ ggü. der „Funktion“ bevorzugt.

**Angreifermodellierung** Die Sicherheitsdefinition für ein SMC-Protokoll besteht aus der idealen Funktionalität, die es in der realen Welt emuliert, und den Annahmen über die Mächtigkeit des Angreifers [41]. Dabei machen wir keine Annahmen über die *Vorgehensweise* des letzteren, sondern nur über seine Angriffskapazitäten.

Die erste Annahme betrifft die Korruptionsstrategie: *statische* Angreifer korrumpieren eine feste Untermenge der Parteien, und der Rest bleibt ehrlich für die gesamte Protokolllaufzeit; *adaptive* Angreifer können dagegen während der Ausführung zusätzliche Parteien korrumpieren, je nachdem was sie zur Laufzeit über sie lernen.

Die zweite Annahme betrifft das erlaubte Verhalten des Angreifers:

- **Passive** bzw. semiehrliche (engl. *semi-honest*) Angreifer interferieren nicht mit der Ausführung des Protokolls, lesen aber den kompletten internen Zustand der korrumpierten Parteien aus und versuchen, daraus vertrauliche Informationen über die ehrlichen Parteien abzuleiten.
- **Verstärkte passive** (engl. *augmented semi-honest*) Angreifer verhalten sich wie die semiehrlichen, dürfen aber zusätzlich die Eingaben der korrumpierten Parteien manipulieren [41, Abschn. 2.2].
- **Aktive** bzw. boshafte (engl. *malicious*) Angreifer können die korrumpierten Parteien dazu anweisen, beliebig vom Protokoll abzuweichen, um seine Ausführung direkt zu beeinflussen [41, Abschn. 2.3].
- **Heimliche** (engl. *covert*) Angreifer können ebenfalls beliebig vom Protokoll abweichen, werden dabei von den ehrlichen Parteien mit einer quantifizierbaren Wahrscheinlichkeit „erwischt“ [41, Abschn. 2.4].

Die dritte Annahme betrifft die Rechenkapazität des Angreifers. Grundsätzlich unterscheiden wir zwischen rechnerisch *unbeschränkten* und *effizienten* Angreifern. Die letzteren müssen in PPT laufen, die ersteren unterliegen keinen solchen Beschränkungen. Daraus ergeben sich die Modelle der informationstheoretischen (perfekten) und der komplexitätstheoretischen (asymptotischen) Sicherheit [41].

In der sicheren Signalverarbeitung werden i. d. R. statische semiehrliche PPT-Angreifer angenommen [49], was aber das schwächste der vorgestellten Modellen ist [52].

**Passive Sicherheit** Um im Simulationsparadigma zu beweisen, dass ein Protokoll $\pi$ die Funktionalität $f$ sicher in Anwesenheit der semiehrlichen (passiven) Angreifern umsetzt, müssen wir zeigen, dass die Parteien *nicht mehr* zusätzliche Information aus seiner Ausführung in der realen Welt lernen, als sie aus der Kommunikation mit der vertrauenswürdigen dritten Partei in der idealen lernen würden. Wir nennen den „Angreifer“ in der idealen Welt „Simulator“, weil er für unseren Beweis aus den Ein-

und Ausgaben der korrumpierten Parteien in der *idealen* Welt ihre gesamte Sicht („View“) auf das *reale* Protokoll $\pi$ so simulieren muss, dass kein PPT-Algorithmus das simulierte Protokolltranskript von einem realen unterscheiden kann.

Um dies für Zweiparteiprotokolle zu formalisieren, übernehmen wir die Notation aus [52, Absch. 6.4.2]:

- Die View der Partei $P_j$ (mit $j \in \{1, 2\}$) während der Ausführung des Protokolls $\pi$ mit den Eingaben $x, y$ und dem Sicherheitsparameter $n$ bezeichnen wir mit $\text{view}_j^\pi(x, y, n)$. Sie besteht aus $\left(w, r^j; m_1^j, \ldots, m_T^j\right)$, wobei $w \in \{x, y\}$ die Eingabe von $P_j$ ist, $r^j$ die Inhalte ihres internen Zufallsbands enthält, $m_t^j$ die $t$. Nachricht repräsentiert, die sie im Protokoll erhalten hat.
- Die Ausgabe der Partei $P_j$ während der Ausführung des Protokolls $\pi$ mit den Eingaben $x, y$ (wie oben definiert) und dem Sicherheitsparameter $n$ bezeichnen wir mit $\text{output}_j^\pi(x, y, n)$. Die gesamte Protokollausgabe bezeichnen wir mit

$$\text{output}^\pi(x, y, n) = \left(\text{output}_1^\pi(x, y, n), \text{output}_2^\pi(x, y, n)\right).$$

**Definition 2.7 (Passive Sicherheit)** Bezeichne $f = (f_1, f_2)$ eine Funktionalität. Wir sagen, dass ein Protokoll $\pi$ die Funktionalität $f$ sicher in der Anwesenheit der statischen semiehrlichen PPT-Angreifern berechnet, wenn solche PPT-Algorithmen $\mathcal{S}_1, \mathcal{S}_2$ existieren, dass gilt:

$$\{\mathcal{S}_1(1^n, x, f_1(x, y)), f(x, y)\}_{x,y,n} \overset{\mathsf{c}}{\equiv} \{\mathsf{view}_1^\pi(x, y, n), \mathsf{output}^\pi(x, y, n)\}_{x,y,n}$$
$$\{\mathcal{S}_2(1^n, y, f_2(x, y)), f(x, y)\}_{x,y,n} \overset{\mathsf{c}}{\equiv} \{\mathsf{view}_2^\pi(x, y, n), \mathsf{output}^\pi(x, y, n)\}_{x,y,n},$$

wobei $x, y \in \{0, 1\}^*$, $|x| = |y|$ und $n \in \mathbb{N}$. (Zitiert nach [52, Def. 6.4.1].)

Weil die Views der Parteien in den realen Protokollen grundsätzlich probabilistisch sind, verlangt die Def. 2.7 nicht, dass die Ausgabe von $\mathcal{S}_j$ *gleich* $\text{view}_j^\pi$ ist, sondern dass die *Verbundverteilung* der Ausgabe von $\mathcal{S}_j$ (gegeben die Ein- und Ausgabe von $P_j$) und der Ausgabe der idealen Funktionalität $f$ (gegeben die Eingaben beider Parteien) *komplexitätstheoretisch ununterscheidbar* von der Verbundverteilung der View von $P_j$ auf das Protokoll und seiner Gesamtausgabe ist.

Die vollständige Formulierung der Sicherheit nach [52, Def. 6.4.1] ist notwendig für Funktionalitäten, deren Ausgaben probabilistisch sind. Für deterministische Funktionalitäten (z. B. Kalman-Filter) ist allerdings eine einfachere Formulierung der passiven Sicherheit möglich, die die Sicherheitsbeweise deutlich vereinfacht.

**Definition 2.8 (Passive Sicherheit für deterministische Funktionalitäten)** Wir sagen, dass ein Protokoll $\pi$ eine deterministische Funktionalität $f$ sicher in der Anwesenheit

der statischen semiehrlichen PPT-Angreifern berechnet, wenn es sowohl *korrekt*, als auch *vertraulich* ist.

**Korrektheit** bedeutet, dass eine solche vernachlässigbare Funktion $\mu(n)$ existiert, dass für jede Eingabe $x, y \in \{0,1\}^*$ und jeden Sicherheitsparameter $n$ gilt:

$$\mathrm{P}\left[\mathsf{output}^{\pi}(x,y,n) \neq f(x,y)\right] \leq \mu(n).$$

**Vertraulichkeit** bedeutet, dass die View jeder Partei (unabhängig voneinander) simuliert werden kann, d. h. es existieren PPT-Algorithmen $\mathcal{S}_1, \mathcal{S}_2$, sodass:

$$\{\mathcal{S}_1(1^n, x, f_1(x,y))\}_{x,y,n} \overset{\mathsf{c}}{\equiv} \{\mathsf{view}_1^{\pi}(x,y,n)\}_{x,y,n}$$
$$\{\mathcal{S}_2(1^n, y, f_2(x,y))\}_{x,y,n} \overset{\mathsf{c}}{\equiv} \{\mathsf{view}_2^{\pi}(x,y,n)\}_{x,y,n}.$$

(Zitiert nach [52, Abschn. 6.4.2].)

Die vereinfachte Formulierung beruht darauf, dass der Unterscheider die Indizes $x, y$ kennt und damit den Wert von $f(x,y)$ für deterministische Funktionalitäten selbst berechnen kann. Weil dieser dank der Korrektheit von $\mathrm{output}^{\pi}(x,y,n)$ ununterscheidbar ist, sind die Def. 2.7 und 2.8 in diesem Spezialfall äquivalent.

**Komponierbarkeit** Oft wollen wir die bereits als sicher bewiesenen Protokolle als Bausteine in größeren Protokollen wiederverwenden. Der kryptographisch korrekte Ansatz dazu ist über die sog. „Vertraulichkeitsreduktionen“ (engl. *privacy reduction*) und „Kompositionssätze“ (engl. *composition theorem*) [35, Abschn. 7.3].

Wir stellen ein Hybridmodell auf, indem die Parteien, wie in der realen Welt, ein Protokoll $\pi_{g|f}$ ausführen, um die Funktionalität $g$ zu berechnen, dabei aber den Zugriff auf ein sog. „Orakel“ (d. h. eine unkorrumpierbare dritte Partei aus der idealen Welt) haben, das sie gemeinsam aufrufen können, um die Funktionalität $f$ berechnen zu lassen. Dabei ist wichtig, dass alle Parteien ihre Eingaben für $f$ gleichzeitig an das Orakel übergeben und keine Berechnungen durchführen, während es läuft, und v. a. dass keine zwei Orakelaufrufe parallel stattfinden. Wir nennen diesen Aufbau die „sequentielle Komposition“ (engl. *sequential composition*).

**Definition 2.9 (Vertraulichkeitsreduktion)** Wir sagen, dass ein orakelunterstütztes Protokoll $\pi_{g|f}$ die Funktionalität $g$ *vertraulich auf* $f$ *reduziert*, wenn es die Orakelfunktionalität $f$ wie oben beschrieben benutzt, um die Funktionalität $g$ sicher in der Anwesenheit der statischen semiehrlichen PPT-Angreifern (nach der Def. 2.7) zu berechnen. (Umformuliert aus [35, Def. 7.3.1 und 7.3.2].)

Der Kompositionssatz für das semiehrliche Modell (Satz 7.3.3 in [35] für zwei Parteien bzw. Satz 7.5.7 für Mehrparteienberechnungen) besagt, dass wenn eine Funktionalität $g$ auf $f$ nach der Def. 2.9 vertraulich reduzierbar ist und ein Protokoll existiert, das $f$ nach der Def. 2.7 vertraulich berechnet, dann existiert auch ein Protokoll, das $g$ vertraulich berechnet. In anderen Worten, wenn wir im Protokoll $\pi_{g|f}$ aus dem Hybridmodell alle Orakelaufrufe durch die Ausführungen des passiv sicheren $\pi_f$ ersetzen, ist das resultierende Protokoll $\pi_g$ ebenfalls passiv sicher.

Die Sicherheit unter dem Kompositionssatz beruht auf einem Hybridargument. Angenommen, wir haben für $\pi_{g|f}$ einen Simulator $\mathcal{S}_j^{g|f}$ und für $\pi_f$ einen $\mathcal{S}_j^f$, die die entsprechenden Views der $P_j$ ununterscheidbar von der realen Welt simulieren. Dann können wir eine Hybridverteilung erzeugen, indem wir die reale $\text{view}_j^{g|f}$ für jeden darin enthaltenen Orakelaufruf $a \leftarrow f(e)$ um die Ausgabe vom $\mathcal{S}_j^f(1^n, e, a)$ ergänzen. Dass diese Hybridverteilung nicht von der realen $\text{view}_j^g$ unterscheidbar ist, folgt daraus, dass $\mathcal{S}_j^f$ die $\text{view}_j^f$ ununterscheidbar simuliert. Analog schließen wir, dass die Ausgabe von $\mathcal{S}_j^g$, in der die Ausgabe von $\mathcal{S}_j^{g|f}$ für jeden Orakelaufruf um die von $\mathcal{S}_j^f$ (entsprechenden parametriert) ergänzt wird, von der $\text{view}_j^g$ ununterscheidbar ist, was uns letztendlich die passive Sicherheit des Protokolls $\pi_g$ herleiten lässt.

Wir bemerken aber, dass diese Ergebnisse im Allgemeinen nur für die sequentielle Komposition gelten. Unter der parallelen Komposition, wenn mehrere Orakelaufrufe bzw. Subprotokollausführungen nebeneinander stattfinden, impliziert die Vertraulichkeit der Subprotokolle nicht mehr die Sicherheit der Gesamtkomposition (das erste Gegenbeispiel hierzu wurde in [36] veröffentlicht). In der Kryptographie wird die parallele Komposition aber kaum gesondert betrachtet – stattdessen wird aktuell aktiv an dem Modell der sog. „universellen Komponierbarkeit“ (engl. *universal composability*, kurz: UC) [16] geforscht, in dem Protokolle als unter beliebiger (auch paralleler) Komposition sicher bewiesen werden können [35, Abschn. 7.7.2]. Weil UC weit außerhalb des Umfangs dieser Arbeit liegt, gehen wir darauf nicht näher ein.

## 2.3 Quantisierung und Fixpunktarithmetik

Um die Signalverarbeitung und die formale Kryptographie letztendlich zusammen zu bringen, benötigen wir zwei weitere Bausteine: Quantisierung und Fixpunktarithmetik. Diese Notwendigkeit ergibt sich aus dem fundamentalen Unterschied in den Rechendomänen der beiden Disziplinen: während die Signalverarbeitung und speziell die Zustandsschätzung mit reellen bzw. als Gleitkommazahlen kodierten Werten rechnet, sind die Haupteinheiten der Kryptographie ein Bit bzw. eine Bitfolge.

**Quantisierung** Das Paillier-Schema unterstützt nativ die homomorphe Addition der Chiffrate mit Klartexte aus $\mathbb{Z}_s \equiv \mathbb{Z}/s\mathbb{Z}$. Um reelle bzw. Gleitkommawerte zu verschlüsseln bzw. zu verarbeiten, kodieren wir diese im Folgenden als Elemente von $\mathbb{Z}_s$, sodass ihre niedrigstwertigen Bits die Dezimalstellen repräsentieren, und verarbeiten sie anschließend nach den Regeln der Fixkommaarithmetik [56]. Dieses Vorgehen wird auch in anderen Arbeiten zur sicheren verteilten Zustandsschätzung mithilfe der homomorphen Verschlüsselung verwendet, z. B. in [38, 67].

**Definition 2.10 (Quantisierung und Rückquantisierung)** Die Quantisierung eines reellen Wertes in ein Integer mit einer Bitpräzisionsstufe $f \in \mathbb{N}$ ist eine Transformation $Quantize : \mathbb{R} \to \mathbb{Z}$:

$$Quantize(x) = \lfloor 2^f \cdot x \rceil.$$

Die Rückquantisierung ist eine Transformation $Unquantize : \mathbb{Z} \to \mathbb{R}$:

$$Unquantize(x) = \frac{x}{2^f}$$

Die Quantisierung und die Rückquantisierung von Arrays (Vektoren und Matrizen) erfolgt elementweise unter der Erhaltung deren Struktur.

Hier und im Folgenden bezeichnen wir mit $\lfloor \cdot \rceil$ die Rundungsoperation zu der nächsten ganzen Zahl, $\lfloor \cdot \rfloor$ die Abrundung zur nächstkleineren und $\lceil \cdot \rceil$ die Aufrundung zur nächstgrößeren Ganzzahl. Offensichtlich ist $2^{-(f+1)}$ die obere Schranke des absoluten Fehlers, der durch die Quantisierung nach der Def. 2.10 entstehen kann.

**Negative Werte** Neben den Dezimalstellen müssen wir oft auch negative Werte verschlüsseln und verarbeiten. Dazu könnten wir das Vorzeichen getrennt kodieren, allerdings ist in der modularen Arithmetik eine viel elegantere Lösung möglich, denn für alle $a \in \mathbb{Z}_s$ gilt:

$$-a \equiv s - a \pmod{s}.$$

**Definition 2.11 (Verschlüsselung negativer Zahlen)** Um Klartexte aus dem Intervall $\left(-\frac{s}{2}, \frac{s}{2}\right)$ statt aus $[0, s)$ im Paillier-Schema mit einem Klartextmodulus $s$ zu verschlüsseln, wird das Chiffrat wie folgt berechnet:

$$c = \begin{cases} \mathsf{Enc}(pk, m), & \text{wenn } m \geq 0 \\ \mathsf{Enc}(pk, m + s), & \text{wenn } m < 0 \end{cases}$$

Die Entschlüsselung erfolgt in diesem Fall analog:

$$m = \begin{cases} \mathsf{Dec}(sk, c), & \text{wenn } \mathsf{Dec}(sk, c) < \frac{s}{2} \\ \mathsf{Dec}(sk, c) - s, & \text{wenn } \mathsf{Dec}(sk, c) > \frac{s}{2} \end{cases}$$

Wir beobachten, dass weil das Klartextmodulus $s$ im Paillier-Schema ein Produkt zweier Primzahlen ist (und keine davon gleich 2 ist), $s$ trivialerweise ungerade und $\frac{s}{2}$ somit keine ganze Zahl ist. Daraus folgt, dass die größte verschlüsselbare Zahl nach der Def. 2.11 gleich $\lfloor \frac{s}{2} \rfloor$ und die kleinste, $-\lfloor \frac{s}{2} \rfloor$ ist, d.h. der modifizierte Nachrichtenraum $\mathcal{M}$ ist um 0 symmetrisch und enthält genauso viele positive, als negative Elemente. Unter der Annahme, dass $s$ über 2000 Bit lang ist, ist $\mathcal{M}$ groß genug, um alle üblichen Floating-Point-Präzisionsstufen nach IEEE 754 [42] darin abzubilden.

**Alternativen** In der Literatur findet sich mindestens ein Vorschlag [29] zur Umsetzung der ausgelagerten Gleitkommaarithmetik nach dem IEEE 754-Standard [42]. Er basiert zu einem großen Teil auf dem *Garbled Circuits*-Prokoll von Andrew Yao und kam deshalb in der vorliegenden Arbeit nicht zum Einsatz. Das Yao-Protokoll [77] ist eine weitere Methode zur kryptographisch sicheren verteilten Berechnung beliebiger Funktionalitäten. Weil die Erklärung seiner Funktionsweise aber sehr aufwendig ist und außerhalb des Umfangs dieser Arbeit liegt, verweisen wir die interessierten Leser auf [41, Kap. 3].

## 2.4 Aktuelle Herausforderungen

Die größte aktuelle Herausforderung auf dem Gebiet der sicheren Signalverarbeitung ist, aus unserer Sicht, das Fehlen eines universell akzeptierten Verständnisses der formalen Sicherheit. Es äußert sich am Häufigsten darin, dass Verfahren und Protokolle entworfen und als sicher erklärt werden, ohne sie aus der Sicht der kryptographisch sicheren Mehrparteienberechnung zu betrachten. Ein Teil davon liegt sicherlich an der historischen Kluft zwischen den Forschungsgebieten der Signalverarbeitung, aus der die meisten dieser Verfahren stammen, und der formalen Kryptographie. Aber zum großen Teil fehlt auch ein gemeinsames Zielbild und ein universelles Vorgehen, wie solche Aufgabenstellungen ganzheitlich anzugehen sind.

Wie aus unserer kurzen Einführung in die moderne Kryptographie eindeutig hervorgeht, ist die beweisbare Sicherheit immer mit zusätzlichem Aufwand verbunden, sowohl beim Protokolldesign, als auch zur Laufzeit. Wir benötigen z. B. im Paillier-Schema ein über 500 Byte langes Chiffrat, um eine 32 Bit Zahl in der verschlüsselten Domäne abzubilden (s. Seite 19). Aus diesem Grund erfordert die sichere Signalverarbeitung pragmatische Kompromisse zwischen den Genauigkeits-, Sicherheits- und Komplexitätsanforderungen in jedem konkreten Anwendungsfall.

Im vorherigen Kapitel gingen wir auf den grundlegenden Domänenunterschied zwischen den Gebieten der Kryptographie und der Signalverarbeitung ein. Die meisten

kryptographischen Verfahren verarbeiten nämlich Bitfolgen oder, wie das Paillier-Schema, Integer modulo große Semiprimzahlen, während die Ein- und Ausgaben der Signalverarbeitungsverfahren normalerweise Vektoren von reellen bzw. Gleitkommazahlen sind. Mit der Quantisierung der Daten vor der Verschlüsselung bringen wir die zwei Felder zusammen, jedoch muss in jedem Protokoll untersucht werden, welche Effekte sie auf seine Genauigkeit hat.

Unser primäres kryptographisches Werkzeug in dieser Arbeit ist die homomorphe Verschlüsselung. Wir haben bereits gezeigt, dass die homomorphe Verformbarkeit der Chiffrate zwar keine Auswirkung auf die IND-CPA-Sicherheit der Verschlüsselungsschemas hat, aber auch dass dadurch die höchste Stufe der Verschlüsselungssicherheit per Definition unerreichbar wird. Eine Herausforderung ist deshalb, präzise Sicherheitsgarantien auszuformulieren, die mithilfe dieses Werkzeugs erreichbar sind, und ggf. ihre inhärenten Schwachstellen mit geeigneten anderen Werkzeugen zu kompensieren.

Homomorphe Berechnungen bringen auch erhebliche funktionelle Einschränkungen mit sich: Operationen wie die (reelle) Division sind in der verschlüsselten Domäne nicht definiert, und Vergleiche von verschlüsselten Zahlen im Allgemeinen unmöglich, weil sie die IND-CPA-Sicherheit des jeweiligen Schemas zunichtemachen würden. Des Weiteren, weil vollhomomorphe Schemas nach wie vor viel rechenintensiver als die teilhomomorphen sind, werden wir in dieser Arbeit uns hauptsächlich mit den letzteren befassen und damit in Kauf nehmen müssen, dass uns grundsätzlich nur eine homomorphe Operation (Addition) zur Verfügung steht. Eine Herausforderung ist also, Filterverfahren in der verschlüsselten Domäne zu entwerfen, die nur mit derart limitierten homomorphen Operationen berechenbar sind.

Aber neben den zahlreichen Schwierigkeiten, die dem Zusammenkommen der Kryptographie und der Signalverarbeitung im Weg stehen, besteht dabei auch das Potenzial, dass zwischen diesen Feldern noch unerforschte Synergien zu entdecken sind. Die große Herausforderung dabei ist es, diese Synergien rechtzeitig zu erkennen und für praxisrelevante Aufgaben auszunutzen.

KAPITEL 3

# Sicheres Extended Kalman-Filter

Der erste Vorschlag zur Umsetzung des Kalman-Filterverfahrens in der verschlüsselten Domäne ist das EKF mit vertraulichen Messeingaben von Gonzalez-Serrano *et al.* [38]. Der Aufbau ihres Protokolls wird im Abschnitt 3.1 zusammengefasst und im Rest dieses Kapitels in Bezug auf seine Sicherheit untersucht.

Zuerst zeigen wir im Abschnitt 3.2, dass durch die Protokollausführung ein systemischer Rundungsfehler entsteht, der im Originalpaper nicht beachtet wird, aber erhebliche Implikationen für die formale Korrektheit des Protokolls hat. Im Abschnitt 3.3 stellen wir eine intuitive Sicherheitsdefinition für das vertrauliche Kalman-Filter auf und zeigen im 3.4, dass das Protokoll sie nicht erfüllt. Anschließend erörtern wir, welche Veränderungen am Protokoll, aber auch an unserem Modell notwendig wären, um seine Sicherheit zu beweisen, und bewerten das Endergebnis im Abschnitt 3.5.

## 3.1 Protokoll

Die Autoren von [38] beschreiben ein Protokoll $\pi_{\text{EKF}}$ mit zwei Parteien, indem der Sensor $P_1$ Messungen der Position des Agenten $P_2$ erhebt, der sie zur eigenen Lokalisierung verwendet. Sie verwenden das additiv homomorphe Paillier-Schema und eine Reihe zusätzlicher Prozeduren, um die fehlenden arithmetischen Operationen damit umzusetzen, die sie für den Kalman-Filterschritt benötigen. Das Protokoll ist im Originalpaper sehr ausführlich beschrieben, deshalb geben wir es hier nicht vollständig wieder, sondern gehen punktuell auf seine wichtigsten Aspekte ein.

**Quantisierung** Alle reellen bzw. Gleitkommazahlen im Protokoll werden zu Fixkommazahlen mit $n_i$ Bits vor und $n_f$ Bits nach dem Komma nach der Def. 2.10 quantisiert, sodass die Gesamtlänge (mit dem Vorzeichenbit) $l = n_i + n_f + 1$ ist. Auf die Kodierung der negativen Zahlen gehen die Autoren nicht näher ein.

**Interaktive quasi-homomorphe Multiplikation** Weil im Protokoll kein vollhomomorphes Verschlüsselungsschema eingesetzt wird, mussten die Autoren ein Subprotokoll entwerfen, mit dem der Agent zwei homomorph verschlüsselte Zahlen $[\![x]\!], [\![y]\!]$ miteinander multiplizieren kann, basierend auf dem „Blinding"-Verfahren. Im Originalvorschlag [49] zieht der Agent zufällig $r_x, r_y \leftarrow \mathbb{Z}_s$, verschlüsselt sie mit dem $pk$ des Sensors und addiert homomorph die verschlüsselten Faktoren auf. Der Sensor empfängt die Chiffrate $[\![x + r_x]\!], [\![y + r_y]\!]$, entschlüsselt sie, multipliziert die Inhalte, verschlüsselt das Produkt und sendet das frische Chiffrat $[\![z]\!]$ zurück an den Agenten, der seine Blinding-Terme dann homomorph wieder abziehen kann:

$$\begin{aligned} [\![z]\!] &= [\![(x + r_x)(y + r_y)]\!] = [\![xy + xr_y + yr_x + r_x r_y]\!] \qquad (3.1)\\ [\![xr_y]\!] &= [\![x]\!] \odot_{pk} r_y \\ [\![yr_x]\!] &= [\![y]\!] \odot_{pk} r_x \\ [\![xy]\!] &= [\![z]\!] -_{pk} [\![xr_y]\!] -_{pk} [\![yr_x]\!] -_{pk} [\![r_x r_y]\!] \\ &= [\![z - (xr_y + yr_x + r_x r_y)]\!]. \end{aligned}$$

Es ist leicht zu zeigen, dass das Verfahren in $\mathbb{Z}_s$ korrekt ist, selbst wenn $r_x, r_y$ groß genug sind, um einen modularen Überlauf bei der Multiplikation zu verursachen. Allerdings sind $x, y$ im $\pi_{\text{EKF}}$ keine generischen Integer, sondern stehen für quantisierte Fixkommazahlen. Weil in der Fixkommaarithmetik das Komma sich bei der Multiplikation um $n_f$ stellen nach links verschiebt, muss der Sensor den Wert $z = (x + r_x)(y + r_y)$ vor dem Wiederverschlüsseln erst normalisieren, indem er ihn durch $2^{n_f}$ teilt und zum nächsten Integer rundet.

Wegen dieser Normalisierung müssen die Blinding-Terme $r_x, r_y$ in $\pi_{\text{EKF}}$ (im Gegensatz zum Originalvorschlag in [49]) so gewählt werden müssen, dass kein modularer Überlauf im Schritt (3.1) stattfinden kann. Die Autoren gehen nicht näher darauf ein, es ist aber leicht nachzurechnen, dass die beiden Terme dann aus der Menge $\{-q, \ldots, q\}$ gezogen werden sollen, wobei

$$q < \sqrt{\left\lfloor \frac{s}{2} \right\rfloor} - 2^{l-1}, \qquad (3.2)$$

$s$ das Klartextmodulus und $l$ die maximale Bitlänge eines quantisierten Werts ist. Wir bemerken aber, dass dadurch die maskierten Terme $x + r_x, y + r_y$ nicht mehr gleichverteilt in $\mathbb{Z}_s$ sind. Hätten wir vor, die Vertraulichkeit dieses Subprotokolls im Simulationsparadigma zu beweisen, könnten wir diese Terme für den Sensor nicht mehr ohne zusätzlichen Annahmen simulieren.

**Matrixprodukt** Das Multiplizieren von Matrizen und Vektoren reduzieren die Autoren auf die mehrfache Anwendung der interaktiven Skalarmultiplikation (s. oben)

und der homomorphen Addition. Der Rechenaufwand davon ist $O(L^3)$ in der Matrixgröße $L$, und ein Thema für die zukünftige Recherche ist, ob Matrixmultiplikation in der verschlüsselten Domäne darüber hinaus beschleunigt werden kann.

**Division und Matrixinversion** Für die Matrixinversion in der Berechnung des Kalman-Gains (2.2) benötigt der Agent eine Möglichkeit, eine verschlüsselte Zahl durch eine andere zu dividieren. Die Fixkomma-Division ist aber im Paillier-Schema nicht ohne weiteres homomorph umsetzbar, weil sie in $\mathbb{Z}_s$ nicht definiert ist. Um dies zu umgehen, schlagen Gonzalez-Serrano *et al.* ein neues Subprotokoll $\pi_{\text{Div}}$ vor, das auf dem iterativen, selbst-korrigierenden Newton-Raphson-Verfahren basiert. Sie reduzieren seine Sicherheit auf die der interaktiven Multiplikation und des sicheren Vergleichsprotokolls $\pi_{\text{Comp}}$ aus [72]. Dabei beobachten sie, dass ihr Protokoll in den numerischen Simulationen maximal 3 Iterationen bis zur Konvergenz benötigte.

Das Subprotokoll zur Matrixinversion $\pi_{\text{Inv}}$ basiert auf der LR-Zerlegung und reduziert sich auf das $\pi_{\text{Comp}}$, die interaktive Matrizenmultiplikation, das $\pi_{\text{Div}}$ und die homomorphe Addition.

**Das EKF-Protokoll** Das vollständige Protokoll $\pi_{\text{EKF}}$ berechnet interaktiv die Schritte (2.1a), (2.1b), (2.2), (2.3a) und (2.3b) des Kalman-Filters. Die Autoren bezeichnen es als ein *Extended* Kalman-Filter, weil aber der Sensor und der Agent die Linearisierung ihres Mess- bzw. Systemmodells jeweils lokal im Klartext durchführen, unterscheidet sich das Verfahren nicht wesentlich vom linearen Kalman-Filter.

Der Sensor hat die Eingaben $\underline{\overset{\star}{z}}_k, \mathbf{H}_k, \mathbf{C}^z$ und der Agent $\underline{\hat{x}}_k, \mathbf{A}, \mathbf{C}^w$ (s. Abschnitt 2.1.1), wobei $k \in \mathbb{N}$ den aktuellen (diskreten) Zeitpunkt bezeichnet. Das Protokoll $\pi_{\text{EKF}}$ besteht grob aus folgenden Schritten:

0. Die Setup-Phase: Der Sensor generiert ein Paillier-Schlüsselpaar $(pk, sk)$ und sendet $pk$, sowie $[\![\mathbf{C}^z]\!]$, an den Agenten, der seinerseits $\underline{\hat{x}}_0^{\text{e}}, \mathbf{C}_0^{\text{e}}$ initialisiert, $[\![\mathbf{C}_0^{\text{e}}]\!], [\![\mathbf{C}^w]\!]$ berechnet und $k \coloneqq 1$ setzt.
1. Der Agent prädiziert den Zustand $\underline{\hat{x}}_k^{\text{p}}$ lokal im Klartext und berechnet $[\![\mathbf{C}_k^{\text{p}}]\!] = [\![\mathbf{A}\mathbf{C}_k^{\text{e}}\mathbf{A}^{\text{T}} + \mathbf{C}^w]\!]$ mithilfe der interaktiven Matrixmultiplikation und der homomorphen elementweise Matrixaddition.
2. Der Sensor sendet $[\![\underline{\overset{\star}{z}}_k]\!], [\![\mathbf{H}_k]\!]$ an den Agenten.
3. Der Agent berechnet interaktiv, analog Schritt 1: $[\![\mathbf{S}_k]\!] = [\![\mathbf{H}_k\mathbf{C}_k^{\text{p}}\mathbf{H}_k^{\text{T}} + \mathbf{C}^z]\!]$.
4. Der Agent und der Sensor führen untereinander das Subprotokoll $\pi_{\text{Inv}}$ mit $[\![\mathbf{S}_k]\!]$ als Eingabe aus, damit der Agent die verschlüsselte Matrix $[\![\mathbf{S}_k^{-1}]\!]$ erhält.

5. Der Agent berechnet interaktiv, analog Schritt 1:

$$\begin{aligned}[\![\mathbf{K}_k]\!] &= [\![\mathbf{C}_k^{\mathrm{p}}\mathbf{H}_k^{\mathrm{T}}\mathbf{S}_k^{-1}]\!] \\ [\![\underline{\hat{x}}_k^{\mathrm{e}}]\!] &= [\![\underline{\hat{x}}_k^{\mathrm{p}} + \mathbf{K}_k\,(\underline{\overset{\star}{z}}_k - \mathbf{H}_k\underline{\hat{x}}_k^{\mathrm{p}})]\!] \\ [\![\mathbf{C}_k^{\mathrm{e}}]\!] &= [\![\mathbf{C}_k^{\mathrm{p}} - \mathbf{K}_k\mathbf{H}_k\mathbf{C}_k^{\mathrm{p}}]\!]\end{aligned}$$

6. Der Agent lässt sich $[\![\underline{\hat{x}}_k^{\mathrm{e}}]\!]$ vom Sensor entschlüsseln, setzt $k \coloneqq k+1$ und geht zum Schritt 1 zurück.

Eine Diskussion der Sicherheitsgarantien des $\pi_{\mathrm{EKF}}$ folgt im Abschnitt 3.4, wir bemerken aber vorerst, dass die komplette Verschleierung von $\mathbf{C}_k^{\mathrm{e}}$ keine konkreten Sicherheitsvorteile, aber einen erheblichen Zusatzrechenaufwand im Schritt 1 mit sich bringt. Sie führt außerdem zu einem etwas seltsamen Zustand, in dem die Kovarianzmatrix der Schätzung ab $k = 1$ keiner Partei explizit bekannt ist, aber jede Runde als versteckte Eingabe in ihre Ausgaben einfließt.

**Undokumentiertes Subprotokoll** In $\pi_{\mathrm{Div}}, \pi_{\mathrm{Inv}}$ und $\pi_{\mathrm{EKF}}$ wird ein weiteres Subprotokoll verwendet, auf das im Originalpaper nicht näher eingegangen wird. Wir bezeichnen es im Folgenden mit $\pi_{\mathrm{Reveal}}$, weil der Agent sich damit ein beliebiges Chiffrat von dem Sensor entschlüsseln lässt, um u. a. die berechnete Zustandsschätzung im letzten Schritt des $\pi_{\mathrm{EKF}}$ im Klartext zu erhalten. Um zu verhindern, dass auch der Sensor die Inhalte des Chiffrats dabei lernt, wurde im $\pi_{\mathrm{Reveal}}$ vermutlich eine Variante vom homomorphen Blinding aus der interaktiven Multiplikation verwendet: z.B. im $\pi_{\mathrm{EKF}}$ zieht der Agent $\underline{r} \leftarrow \mathbb{Z}_s^N$, sendet $[\![\underline{\hat{x}}_k^{\mathrm{e}} + \underline{r}]\!]$ an den Sensor, bekommt $\underline{\hat{x}}_k^{\mathrm{e}} + \underline{r}$ zurück, und berechnet daraus letztendlich $\underline{\hat{x}}_k^{\mathrm{e}}$ im Klartext.

## 3.2 Rundungsfehleranalyse

Bevor wir zur Sicherheitsmodellierung übergehen, müssen wir den Rundungsfehler betrachten, der durch die interaktive quasi-homomorphe Multiplikation entsteht, weil dieser später wichtige Implikationen für die Sicherheit haben wird. Um die Herleitung leserlicher zu gestalten, betrachten wir o. B. d. A. ein äquivalentes ideales Protokoll in der reellen Domäne ohne Quantisierung oder Verschlüsselung. Der Agent hat darin die Eingaben $x, y \in \mathbb{R}$ und der Sensor, die leere Eingabe.

1. Der Agent zieht $u_x, u_y \leftarrow \{-q, \ldots, q\}$, wobei für $q$ die Formel (3.2) gilt, und berechnet die Blinding-Terme $\boldsymbol{r}_x = 2^{-n_f} \cdot u_x, \boldsymbol{r}_y = 2^{-n_f} \cdot u_y \in \mathbb{R}$.
2. Der Agent berechnet $\boldsymbol{x}' = x + \boldsymbol{r}_x, \boldsymbol{y}' = y + \boldsymbol{r}_y$ und sendet sie an den Sensor.
3. Der Sensor berechnet $\boldsymbol{z}' = \boldsymbol{x}'\boldsymbol{y}'$ und sendet $\boldsymbol{z}'$ an den Agenten.

4. Der Agent berechnet den Korrekturterm $\boldsymbol{k} = x\boldsymbol{r}_y + y\boldsymbol{r}_x + \boldsymbol{r}_x\boldsymbol{r}_y$.

5. Der Agent berechnet $z = \boldsymbol{z}' - \boldsymbol{k}$ und gibt $z$ aus.

In dieser Formulierung ist das Protokoll offensichtlich korrekt, denn $z = xy$ für alle $x, y$. Wir müssen aber beachten, dass $x, y$ dem Agenten nur verschlüsselt vorliegen und er sie deshalb nur mit ganzen Zahlen homomorph multiplizieren kann. Um den Korrekturterm im Schritt 4 zu berechnen, kann er die Chiffrate $[\![x]\!], [\![y]\!]$ weder mit $u_y, u_x$ (weil $xu_y \neq x\boldsymbol{r}_y$ und $yu_x \neq y\boldsymbol{r}_x$), noch mit $\boldsymbol{r}_y, \boldsymbol{r}_x$ (weil diese im Allgemeinen bis zu $n_f$ Nachkommastellen haben) multiplizieren. Er kann also nicht $[\![x\boldsymbol{r}_y]\!]$ und $[\![y\boldsymbol{r}_x]\!]$, sondern nur näherungsweise $[\![x\lfloor\boldsymbol{r}_y\rceil]\!]$ bzw. $[\![y\lfloor\boldsymbol{r}_x\rceil]\!]$ berechnen, indem er $\boldsymbol{r}_y, \boldsymbol{r}_x$ für die homomorphe Multiplikation zur nächsten ganzen Zahl rundet. Für den Rundungsfehler $\boldsymbol{e}_z$, der bei der Berechnung von $z$ mit dem auf dieser Weise approximierten Korrekturterm $\boldsymbol{k}' = x\lfloor\boldsymbol{r}_y\rceil + y\lfloor\boldsymbol{r}_x\rceil + \boldsymbol{r}_x\boldsymbol{r}_y$ entsteht, gilt:

$$\begin{aligned}
\boldsymbol{e}_z &= z - xy = \boldsymbol{z}' - \boldsymbol{k}' - xy \\
&= (x + \boldsymbol{r}_x)(y + \boldsymbol{r}_y) - (x\lfloor\boldsymbol{r}_y\rceil + y\lfloor\boldsymbol{r}_x\rceil + \boldsymbol{r}_x\boldsymbol{r}_y) - xy \\
&= xy + x\boldsymbol{r}_y + y\boldsymbol{r}_x + \boldsymbol{r}_x\boldsymbol{r}_y - x\lfloor\boldsymbol{r}_y\rceil - y\lfloor\boldsymbol{r}_x\rceil - \boldsymbol{r}_x\boldsymbol{r}_y - xy \\
&= x\underbrace{(\boldsymbol{r}_y - \lfloor\boldsymbol{r}_y\rceil)}_{\boldsymbol{\varepsilon}_y} + y\underbrace{(\boldsymbol{r}_x - \lfloor\boldsymbol{r}_x\rceil)}_{\boldsymbol{\varepsilon}_x}.
\end{aligned}$$

Weil $\boldsymbol{r}_x, \boldsymbol{r}_y$ unabhängig und gleichverteilt aus ihrem entsprechenden Intervall gezogen wurden, sind auch die Werte von $\boldsymbol{\varepsilon}_x, \boldsymbol{\varepsilon}_y$ auf dem Intervall $[-0.5, 0.5)$ gleichverteilt. Daraus folgt, dass $\boldsymbol{e}_z$ auf dem Intervall $\left[-\frac{x+y}{2}, \frac{x+y}{2}\right)$ verteilt ist, aber zu $\mathrm{E}\{\boldsymbol{e}_z\} = 0$ tendiert. Intuitiv bedeutet es, dass das Ergebnis $z$ jeder interaktiven quasi-homomorphen Multiplikation um einen Fehlerterm in der Größenordnung der beiden Faktoren $x, y$ vom tatsächlich gesuchten Produkt $xy$ abweichen kann.

Dieser Fehler ist den Technologiebeschränkungen geschuldet und kann ohne weitere Abschwächung der Sicherheitsgarantien nicht vermieden werden: wir könnten z. B. die Blinding-Terme so ziehen, dass $(\boldsymbol{r}_x - \lfloor\boldsymbol{r}_x\rceil) = (\boldsymbol{r}_y - \lfloor\boldsymbol{r}_y\rceil) = 0$, dann würden sie aber keine Dezimalstellen mehr von $x, y$ vor dem Sensor verstecken. Die Alternative wäre ein Technologiewechsel, z. B. von der teil- zu der vollhomomorphen Verschlüsselung, dieser würde aber eine komplette Neukonzeption des Protokolls erfordern.

Der oben hergeleitete Fehler gilt für die interaktive Skalarmultiplikation, die im $\pi_{\mathrm{EKF}}$ aber fast ausschließlich zusammen mit der homomorhen Addition in der Matrixmultiplikation verwendet wird. Dies bedeutet, dass die Fehlerterme aus einzelnen Skalarmultiplikationen sich homomorph aufsummieren und teilweise aufheben, sodass der Gesamtfehler aus der Multiplikation in der Größe der multiplizierten Matrizen gegen 0 strebt, sich aber nie komplett eliminieren lässt.

## 3.3 Sicherheitsmodell

Gonzalez-Serrano *et al.* schlagen in ihrem Paper keine formale Sicherheitsdefinition für ihr Protokoll vor, außer dass es im semiehrlichen (passiven) Angreifermodell nach [35] sicher sein soll. Wir gehen davon aus, dass sie dabei den statischen semiehrlichen PPT-beschränkten Angreifer meinen, der *de facto* das Standard in der Literatur zur sicheren Signalverarbeitung ist [49].

Weil das Originalpaper weder eine Formulierung der idealen Funktionalität enthält, die das Protokoll sicher implementieren soll, noch ein Sicherheitsspiel wie IND-CPA definiert, das ein Angreifer gewinnen muss, um es zu brechen, müssen wir eine eigene ideale Funktionalität vorschlagen und das Protokoll $\pi_{\mathrm{EKF}}$ ihr gegenüber stellen.

**Definition 3.1 (Ideales Kalman-Filter mit vertraulichen Messungen)** Wir definieren eine ideale Funktionalität zum Berechnen des Kalman-Filters:

$$f_{\mathsf{KF}}\left((\overset{\star}{\underline{z}}_1, \mathbf{H}_1, \ldots, \overset{\star}{\underline{z}}_K, \mathbf{H}_K, \mathbf{C}^z), (\hat{\underline{x}}_0^{\mathsf{e}}, \mathbf{C}_0^{\mathsf{e}}, \mathbf{A}, \mathbf{C}^w)\right) = \left(\lambda, (\hat{\underline{x}}_1^{\mathsf{e}}, \ldots, \hat{\underline{x}}_K^{\mathsf{e}})\right), \tag{3.3}$$

wobei die Rundenanzahl $K = \mathsf{poly}(n) \geq 1$, $\lambda$ für eine leere Ausgabe steht, und die Ausgaben $\hat{\underline{x}}_k^{\mathsf{e}}$ für $k \in \{1, \ldots, K\}$ nach den Formeln (2.1a), (2.1b), (2.2), (2.3a) und (2.3b) berechnet werden.

Intuitiv formalisiert die Funktionalität (3.3) die Wunschvorstellung von Gonzalez-Serrano *et al.* [38, Abschn. III], dass der Agent nichts über das Messmodell und die Messwerte des Sensors lernt (was er nicht aus seinen Schätzungen ausrechnen kann), während der Sensor nichts über das Systemmodell des Agenten erfährt (was er nicht aus seinem eigenen Messsverlauf ausrechnen kann).

Mit $\mathsf{poly}(n)$ bezeichnen wir hier und im Folgenden eine abstrakte Funktion, die höchstens polynomiell im Sicherheitsparameter $n$ wächst. Die Beschränkung der Protokolllaufzeit auf $K$ Zeitrunden impliziert, dass darin nur polynomiell viele Chiffrate ausgetauscht werden, was eine Voraussetzung für die Gültigkeit der komplexitätstheoretischen Annahmen ist, die wir für den Sicherheitsbeweis benötigen.

**Formalisierung der Subprotokolle** In dem Originalpaper besteht $\pi_{\mathrm{EKF}}$ aus mehreren Subprotokollen, die sequentiell komponiert werden. Normalerweise würden wir für alle Subprotokolle, z. B. die interaktive Multiplikation oder Division, jeweils eigene ideale Funktionalitäten formulieren, ihre Sicherheit beweisen und mithilfe des sequentiellen Kompositionssatzes [35, Satz 7.3.3] die Sicherheit des Gesamtprotokolls herleiten. Jedoch haben alle Protokolle von Gonzalez-Serrano *et al.* (außer $\pi_{\mathrm{EKF}}$ selbst) ausschließlich verschlüsselte Ein- und Ausgaben, sodass die Parteien

grundsätzlich kein Wissen darüber haben, was sie eigentlich verarbeiten, und die entsprechende ideale Funktionalität dafür durchgehend $f(\lambda, \lambda) = (\lambda, \lambda)$ wäre. Weil diese Funktionalität trivialerweise immer korrekt und vertraulich ist, müssen wir stattdessen das Kalman-Filterprotokoll als ein großes Ganzes betrachten.

## 3.4 Sicherheitsanalyse

Wir wollen formal beweisen, dass das Protokoll $\pi_{\text{EKF}}$ sicher ist, d. h. die ideale Funktionalität $f_{\text{KF}}$ nach der Def. 3.1 in der Anwesenheit der statischen semiehrlichen PPT-Angreifern sicher berechnet. Weil die Funktionalität $f_{\text{KF}}$ deterministisch ist, können wir für diesen Beweis die vereinfachte Sicherheitsformulierung nach der Def. 2.8 verwenden. Wir werden aber sehen, dass in diesem Fall weder die Korrektheit, noch die Vertraulichkeit formal gegeben sind, weil das $\pi_{\text{EKF}}$ die $f_{\text{KF}}$ tatsächlich *nicht* sicher umsetzt.

**Korrektheit** Die formale Korrektheit verlangt, dass die Ausgabe des Protokolls $\pi_{\text{EKF}}$ mit einer höchstens in $n$ vernachlässigbaren Wahrscheinlichkeit von der Ausgabe der idealen Funktionalität $f_{\text{KF}}$ abweicht. Jedoch können wir leicht nachvollziehen, dass die Wahrscheinlichkeit einer solchen Abweichung tatsächlich überwältigend ist.

Wie bereits im Abschnitt 3.2 gezeigt wurde, entsteht bei der interaktiven quasihomomorphen Multiplikation ein Rundungsfehler in der Größenordnung der beiden Faktoren. Dieser ist zwar um 0 verteilt und deshalb in der Matrixmultiplikation zu 0 tendiert, resultiert aber mit einer überwältigenden Wahrscheinlichkeit in der Verfälschung der Ausgaben. Da dieser Fehler unabhängig vom Quantisierungsfehler der Faktoren entsteht und sich nicht komplett eliminieren lässt, schließt er grundsätzlich die Korrektheit des Protokolls $\pi_{\text{EKF}}$ aus, – es sei denn, wir ergänzen unsere ideale Funktionalität um die präzise Definition dieser Fehlerverteilung.

An dieser Stelle könnten wir aber auch die Frage stellen, ob die formale Korrektheit nach der strengen kryptographischen Definition für die sichere Signalverarbeitung das Richtige ist. Weil unsere Protokolleingaben verrauschte Messungen und die Ausgaben, stochastische Erwartungswerte sind, erheben wir üblicherweise keinen Anspruch auf überwältigende Korrektheit. Aus dieser Sicht benötigt die sichere Signalverarbeitung möglicherweise eine abgeschwächte, aber pragmatischere Korrektheitsdefinition, die die „ungefähre“ Genauigkeit formalisiert. Wir schlagen in dieser Arbeit allerdings keine solche Definition vor, weil alle weiter im Text vorgestellte Verfahren bereits die formale Korrektheit erfüllen.

**Vertraulichkeit** Wenn das Protokoll die ideale Funktionalität nicht korrekt berechnet, kann grundsätzlich auch kein Simulator (ohne weiteren Annahmen) seine reale Ausgabe basierend auf der idealen Ausgabe fälschen. Wir fahren aber trotzdem mit der Analyse der Vertraulichkeit fort, um weitere wichtige Schwachstellen des Protokolls $\pi_{\text{EKF}}$ aufzuzeigen. Um seine Vertraulichkeit zu beweisen, müssten wir solche Simulatoren $\mathcal{S}_1, \mathcal{S}_2$ finden, dass gilt:

$$\begin{aligned}\left\{\mathcal{S}_1\big(1^n, x, f_1^{\text{KF}}(x,y)\big)\right\}_{x,y,n} &\overset{\text{c}}{\equiv} \left\{\text{view}_1^{\pi_{\text{EKF}}}(x,y,n)\right\}_{x,y,n}\\ \left\{\mathcal{S}_2\big(1^n, y, f_2^{\text{KF}}(x,y)\big)\right\}_{x,y,n} &\overset{\text{c}}{\equiv} \left\{\text{view}_2^{\pi_{\text{EKF}}}(x,y,n)\right\}_{x,y,n},\end{aligned}$$

wobei $x = (\overset{\star}{\underline{z}}_1, \mathbf{H}_1, \ldots, \overset{\star}{\underline{z}}_K, \mathbf{H}_K, \mathbf{C}^z)$, $y = (\underline{\hat{x}}_0^{\text{e}}, \mathbf{C}_0^{\text{e}}, \mathbf{A}, \mathbf{C}^w)$ und $f_1^{\text{KF}}(x,y)$ per Def. 3.1 $\lambda$ (leer) ist. Offensichtlich ist $x$ hier viel länger als $y$, was der Voraussetzung $|x| = |y|$ aus der Def. 2.7 widerspricht; weil unsere Eingaben aber binär kodierte Integer sind, können wir $y$ auf die Länge von $x$ mit führenden Nullen padden, ohne die Werte zu verfälschen. Zwecks Übersichtlichkeit schreiben wir die Indizes $x, y, n$ der Wahrscheinlichkeitsensembles in dem Rest der Arbeit nicht weiter aus.

Wir können nachweisen, dass das Protokoll mehr Informationen über die vertrauliche Eingaben der Parteien preisgibt, als formal erlaubt ist, und deshalb keine Simulatoren wie oben beschrieben konstruiert werden können. Dazu betrachten wir genauer die Verwendung in $\pi_{\text{EKF}}$ des sicheren Vergleichsprotokolls $\pi_{\text{Comp}}$. In seiner Originaldefinition [72] sind die Eingaben von $P_1$ zwei homomorph verschlüsselte Zahlen $[\![a]\!], [\![b]\!]$ und die Ausgabe das *homomorph verschlüsselte* Bit $[\![t]\!]$ wobei $t = 1$, wenn $a \leq b$, sonst $t = 0$; $P_2$ hat in dem Protokoll den Geheimschlüssel $sk$ und eine leere Ausgabe. Abgesehen von der bereits diskutierten Problematik der verschlüsselten „Eingaben“, ist $\pi_{\text{Comp}}$ insofern vertraulich, dass weder $P_1$, noch $P_2$ irgendwas über die tatsächlichen Klartextwerte $a, b, t$ lernen.

Gonzalez-Serrano *et al.* verwenden aber die Ausgabe des Agenten im $\pi_{\text{Comp}}$, um den Kontrollfluss in den Subprotokollen $\pi_{\text{Div}}$ (Schritt 3) und $\pi_{\text{Inv}}$ (Schritt 1) zu steuern, was erhebliche Implikationen das Gesamtprotokoll hat. Die offensichtlichste davon ist, dass der Agent das undokumentierte Protokoll $\pi_{\text{Reveal}}$ verwenden muss, um das Ergebnis vom $\pi_{\text{Comp}}$ im Klartext zu erhalten. Damit werden ihm Informationen über die verschlüsselte Zahlen offengelegt, die nicht in der Sicherheitsdefinition 3.1 enthalten sind. Aber auch der Sensor lernt diese Informationen indirekt, weil der Agent in $\pi_{\text{Div}}$ und in $\pi_{\text{Inv}}$ unterschiedlich viele Nachrichten mit dem Sensor austauscht, je nach dem Ergebnis des Vergleichs.

Der Sensor-Simulator $\mathcal{S}_1$ ist grundsätzlich nicht in der Lage, eine vom realen Protokoll ununterscheidbare View zu erzeugen, weil er in der idealen Welt keine Informationen über die Vergleichsergebnisse erhält, und deshalb nicht wissen kann, *wie viele*

Nachrichten der Agent versenden würde. Der Agent-Simulator $\mathcal{S}_2$ kann währenddessen die reale View des Agenten nicht simulieren, weil die View die Ergebnisse der Vergleiche im Klartext (bzw. mit dem eigenen Zufall von $P_2$ maskiert) enthält, die er aus ihm vorliegenden Ein- und Ausgaben im Allgemeinen nicht ausrechnen kann. Damit können die Ausgaben der Simulatoren nicht von den realen Views der Parteien ununterscheidbar sein, was der Vertraulichkeit des Protokolls widerspricht.

**Mögliche Verbesserungen** Wir könnten die Vertraulichkeit des Subprotokolls $\pi_{\text{Div}}$ erreichen, indem wir die Unterprotokolle $\pi_{\text{Comp}}$ und $\pi_{\text{Reveal}}$ daraus entfernen, und stattdessen die `while`-Schleife im Schritt 3 einfach eine feste Anzahl von Runden laufen lassen. Dies ist möglich, weil die Autoren bemerken, dass ihr Verfahren nach maximal drei Runden konvergiert hatte. Weil im modifizierten Protokoll eine konstante Anzahl von verschlüsselten Nachrichten ausgetauscht wird, sind sie für $\mathcal{S}_1$ simulierbar (vorausgesetzt, er kann die geblendeten Terme für die interaktive Multiplikation erfolgreich fälschen), während $P_2$ keine zusätzlichen Informationen über die Chiffrate aus der Protokollausführung gewinnt, was die Simulation seiner View durch $\mathcal{S}_2$ ermöglicht. Wir können dann aber nicht mehr garantieren, dass das Protokoll für alle Eingaben korrekt bleibt.

Auf das $\pi_{\text{Inv}}$ können wir keine analoge Verbesserung anwenden, weil das $\pi_{\text{Comp}}$ darin benutzt wird, um die Division durch Null zu vermeiden. Dabei lernt der Agent aber nicht nur, ob der Inhalt des Teiler-Chiffrats gleich 0 ist, sondern auch sein Vorzeichen. Um das Protokoll vertraulich zu machen, müssen wir einerseits die `then`-Klausel im Schritt 1 um Dummy-Kommunikationen ergänzen, sodass darin genauso viele Nachrichten wie in der `else`-Klausel ausgetauscht werden, damit der Sensor nichts über den Kontrollfluss des Agenten lernt. Andererseits müssen wir aber die Sicherheitsdefinition 3.1 abschwächen und zusätzliche Informationen explizit in die Ausgabe des Agenten in jeder Zeitrunde $k$ hinzufügen, nämlich die Vorzeichen der Elemente auf der Subdiagonale der $[\![\mathbf{S}_k]\!]$ im Schritt 4.

## 3.5 Bewertung

In diesem Kapitel haben wir die Gefahren des Ansatzes aufgezeigt, kryptographische Protokolle zu entwerfen, ohne eine vollständige Sicherheitsdefinition dafür erstellt zu haben. Um formale Sicherheitsgarantien zu geben, reicht es nicht aus, als sicher bewiesene Bausteine miteinander zu verknüpfen, sondern das gesamte Protokoll muss hinsichtlich seiner Simulierbarkeit im Ideal/Real-Paradigma untersucht werden.

Die erste Schwerigkeit bei der Sicherheitsanalyse des Protokolls $\pi_{\text{EKF}}$ war die Un-

vollständigkeit seiner Sicherheitsdefinition. Um es überhaupt untersuchen zu können, mussten wir eine eigene, naheliegende Funktionalität für die verteilte Kalman-Filter-Berechnung in der Def. 3.1 formulieren. Für die darin enthaltene Subprotokolle (inkl. das undokumentierte $\pi_{\text{Reveal}}$) konnte dagegen keinen Funktionalitäten gefunden werden, weil sie verschlüsselte Ein- bzw. Ausgaben haben. Deshalb war es nicht möglich, ihre Sicherheit einzeln zu beweisen und die des Gesamtprotokolls mithilfe der Vertraulichkeitsreduktionen und des Kompositionssatzes daraus herzuleiten.

Bei der interaktiven quasi-homomorphen Multiplikation wird im $\pi_{\text{EKF}}$ ein Verfahren, das in [49] ursprünglich für die modulare Multiplikation in $\mathbb{Z}_s$ entworfen wurde, auf Fixkommazahlen angewendet, ohne dabei aber den resultierenden Rundungsfehler zu betrachten. Im Abschnitt 3.2 haben wir gezeigt, dass dieser Fehler zusätzlich zum Quantisierungsfehler entsteht, in seiner Größe mit den jeweiligen Faktoren vergleichbar ist, und letztendlich die formale Korrektheit vom $\pi_{\text{EKF}}$ ausschließt.

Bei der Division und der Matrixinversion wird in $\pi_{\text{EKF}}$ das Vergleichsprotokoll aus [72] als Baustein verwendet, allerdings benötigt die jeweilige Operation sein Ergebnis im Klartext, um ihren Kontrollfluss damit zu steuern. Im Abschnitt 3.4 haben wird gezeigt, dass beide Parteien dadurch Informationen über die Chiffratinhalte lernen, die ihnen per Sicherheitsdefinition 3.1 nicht zustehen, was letztendlich auch die Vertraulichkeit von $\pi_{\text{EKF}}$ verhindert.

Im Abschnitt 3.4 erörterten wir mögliche Anpassungen, die am Protokoll und an der idealen Funktionalität notwendig wären, um die Sicherheit des Protokolls $\pi_{\text{EKF}}$ im Simulationsparadigma doch noch zu beweisen. Allerdings stellen wir in Frage, wie ideal eine solche „ideale Funktionalität“ und, folglich, wie wertvoll unsere Sicherheitsgarantien dann noch wären. Ein Beweggrund für diesen Beweis wäre, ihn als Vorstufe zur aktiven Sicherheit zu benutzen, es ist aber zweifelhaft, dass das Protokoll irgendeinen Maß an Sicherheit gegen aktive Angreifer anbieten kann, wenn sie sich jederzeit mit dem $\pi_{\text{Reveal}}$ beliebige Chiffrate entschlüsseln lassen könnten.

Wir beobachten aber, dass die meisten Vertraulichkeitsprobleme vom $\pi_{\text{EKF}}$ im Zusammenhang mit der Matrixinversion auftreten. Wenn wir zulassen, dass der Agent seine aktuelle Kovarianzmatrix $\mathbf{C}_k^{\text{e}}$ im Klartext kennt und der Sensor ihm seine Informationsmatrix $\mathbf{I}_k$ nach (2.5b) im Klartext ausliefert, vermeiden wir die quasi-homomorphe Division komplett, weil der Agent dann die $[\![\mathbf{S}_k^{-1}]\!]$ mithilfe des Matrixinversionslemmas nur mit interaktiven Multiplikationen und homomorphen Additionen berechnen könnte. Allerdings kann er dann auch aus ihm vorliegenden Informationen $(\underline{\hat{x}}_k^{\text{p}}, \mathbf{C}_k^{\text{p}}, \underline{\hat{x}}_k^{\text{e}}, \mathbf{C}_k^{\text{e}}, \mathbf{I}_k)$ den Informationsvektor $\hat{i}_k$ des Sensors analytisch berechnen. Das modifizierte Verfahren kann deshalb keine stärkeren Sicherheitsgarantieren geben, als die Ausführung eines verteilten Informationsfilters im Klartext.

KAPITEL 4

# Sicheres verteiltes Informationsfilter

Im letzten Kapitel haben wir gesehen, dass die Matrixinversion, die den Kern des Kalman-Filterschritts (2.2) bildet, eine große Hürde auf dem Weg zu einem formal sicheren Filterprotokoll darstellt, weil die dafür benötigte Fixpunktdivision in der verschlüsselten Domäne nicht definiert ist. Des Weiteren haben wir gesehen, dass die interaktive quasi-homomorphe Multiplikation mit einem erheblichen Rundungsfehler verbunden ist. In diesem Kapitel wollen wir nun die Frage betrachten, ob wir ein Protokoll entwerfen können, indem die Multiplikation und die Matrixinversion lokal im Klartext und nur noch die Addition homomorph berechnet werden müssen, und in welchen Kontexten es einsetzbar wäre.

Im Abschnitt 2.1.2 haben wir bereits das Informationsfilter beschrieben – eine Umformulierung des Kalman-Filters, die in der verteilten Zustandsschätzung eingesetzt wird. Nun stellen wir dafür im Abschnitt 4.1 ein mögliches Anwendungsszenario und eine Sicherheitsdefinition auf, in denen die verteilten Sensoren ihre Messungen vor dem Verschlüsseln in die Informationsform umwandeln. Dann werden die Messungen (homomorph) aggregiert, entschlüsselt und im Klartext mit dem prädizierten Zustand fusioniert.

Im Abschnitt 4.2 beschreiben wir ein Protokoll, das die ideale Funktionalität aus unserer Sicherheitsdefinition realisiert, und stellen dafür einen ausführlichen Sicherheitsbeweis (Abschnitt 4.3) und eine Komplexitätsanalyse (Abschnitt 4.4) auf, um die allgemeine Vorgehensweise zu verdeutlichen. Im Abschnitt 4.5 erklären wir, wie unser Protokoll um zusätzliche Aggregationsstellen („Hubs“) erweitert werden kann, und wie sich die Größe des Sensornetzwerks dabei verheimlichen lässt. Zum Schluss präsentieren wir die Ergebnisse der numerischen Evaluation einer prototypischen Umsetzung unseres Protokolls im Abschnitt 4.6, und bewerten es in Bezug auf alle bisher genannte Aspekte im Abschnitt 4.7.

## 4.1 Sicherheitsmodell

Wie auch im Kapitel 3 fangen wir mit einem sinnvollen Sicherheitsmodell für unser Verfahren an. Zuerst betrachten ein mögliches Szenario: Ein mobiler Agent bewegt sich über einen Bereich, der von $N$ vernetzten Sensoren überwacht wird, und will sich mit deren Hilfe lokalisieren. Dabei kann der Agent sich nicht darauf verlassen, dass keiner der Sensoren von einer dritten Partei korrumpiert und belauscht wird, aber auch der Sensornetzwerkbetreiber möchte nicht, dass der Agent Details über die Netzwerkarchitektur und die Messmodelle der Sensoren erfährt. Des weiteren möchte der Netzwerkbetreiber die Rechenlast möglichst auf die leistungsstarken „Hubs“ auslagern und den Agent nur mit einem öffentlich zugänglichen „Zentralhub“ kommunizieren lassen. Es finden insgesamt $K$ Kommunikationsrunden bzw. Filterschritte statt, bis der Agent den überwachten Bereich wieder verlässt.

**Definition 4.1 (Ideales Informationsfilter)** Wir definieren eine ideale Funktionalität zum Berechnen des verteilten Informationsfilters mit $N$ Eingaben und $N+1$ Parteien:

$$f_{\mathsf{IF}}(E_1, E_2, \ldots, E_N, \lambda) = (\lambda, \lambda, \ldots, \lambda, A_{N+1}),$$

wobei $N = \mathsf{poly}(n) \geq 2$, $\lambda$ für eine leere Ein- bzw. Ausgabe steht, und $E_j$ und $A_{N+1}$ jeweils Tupel aus $K$ Paare aus je einem Informationsvektor und einer Informationsmatrix sind. Für $j \in \{1, \ldots, N\}$ gilt dabei:

$$\begin{aligned} E_j &= \left(\hat{\underline{i}}_1^j, \mathbf{I}_1^j, \hat{\underline{i}}_2^j, \mathbf{I}_2^j, \ldots, \hat{\underline{i}}_K^j, \mathbf{I}_K^j\right) \\ A_{N+1} &= \left(\hat{\underline{i}}_1, \mathbf{I}_1, \hat{\underline{i}}_2, \mathbf{I}_2, \ldots, \hat{\underline{i}}_K, \mathbf{I}_K\right) \\ \hat{\underline{i}}_k &= \sum_{j=1}^{N} \hat{\underline{i}}_k^j \quad \text{und} \quad \mathbf{I}_k = \sum_{j=1}^{N} \mathbf{I}_k^j, \end{aligned} \tag{4.1}$$

wobei $K = \mathsf{poly}(n) \geq 1$, $\hat{\underline{i}}_k^j, \hat{\underline{i}}_k \in \mathbb{R}^L$, $\mathbf{I}_k^j, \mathbf{I}_k \in \mathbb{R}^{L \times L}$, und $L$ die zeitinvariante Länge des Systemzustandsvektors ist.

In anderen Worten betrachten wir hier statt der einzelnen Kommunikationsrunden den kompletten Kommunikationsablauf: die Eingabe jedes der $N$ Sensoren besteht aus seinem jeweiligen gesamten Messverlauf (in Informationsform), während die Ausgabe des Agenten die Sequenz der vollständig aggregierten Informationsvektoren und -matrizen ist. Dies impliziert, dass die individuellen Messungen weder voneinander, noch von den späteren Zustandsschätzungen abhängen, d. h. auch unabhängig vom Schätzungsprozess erhoben werden könnten.

Offensichtlich implementiert $f_{\mathrm{IF}}$ nach der Def. 4.1 nicht das vollständige Informationsfilter, wie es im Abschnitt 2.1.2 beschrieben wurde: es fehlen vor allem die

Prädiktion und die Fusion der Messdaten in den prädizierten Zustand. Diese Schritte werden aus der idealen Funktionalität weggelassen, weil sie lokal beim Agenten berechnet werden können und dadurch keine Kommunikation mit dem Sensornetzwerk erfordern. Es ist nicht schwer zu erkennen, dass die Funktionalität sich in dieser Formulierung auf eine relativ simple Vektor- bzw. Matrixsummenbildung reduziert, was die Beweise stark vereinfacht.

**Angreifermodell** Unser Ziel ist ein Protokoll $\pi_{\text{IF}}$, das die Funktionalität $f_{\text{IF}}$ sicher in der Anwesenheit der statischen semiehrlichen PPT-Angreifern berechnet.

## 4.2 Protokoll

Wir schlagen das Protokoll 4.1 ($\pi_{\text{IF}}$) auf der Seite 42 vor, um die Funktionalität $f_{\text{IF}}$ zu implementieren. Dabei wollen wir eine nicht-kryptographische Randbedingung beachten, dass der Rechenaufwand zur Aggregation der Informationsvektoren auf mehrere dedizierte Sensorparteien („Hubs“) verteilt wird, die ihre eigenen Messungen mit denen ihrer Nachbarn voraggregieren und an den Zentralhub weitergeben.

**Definition 4.2 (Sensornetzwerkarchitektur)** Die Funktion

$$MyHub : \{1, \ldots, N\} \to \{1, \ldots, N+1\}$$

weist jeder Sensor-ID die ID seines jeweiligen Hubs bzw. des Agenten zu. Umgekehrt weist die Funktion

$$MySenders : \{1, \ldots, N+1\} \to \mathcal{P}\left(\{1, \ldots, N\}\right)$$

jeder Hub-ID die IDs der Sensoren zu, die ihm ihre (ggf. voraggregierte) Messungen senden, sodass $s \in MySenders(j)$ gdw. $MyHub(s) = j$.

O. B. d. A. definieren wir eine spezielle Sensornetzwerkarchitektur für unser Protokoll (s. auch die Abb. 4.1). Mit $H \coloneqq \lceil \frac{N}{2} \rceil$ gilt:

$$MyHub(j) = \begin{cases} N+1, & \text{wenn } j = 1 \\ 1, & \text{wenn } j \in \{2, 3\} \\ 2, & \text{wenn } j \in \{4, \ldots, H\} \\ 3, & \text{wenn } j \in \{H+1, \ldots, N\} \end{cases}$$

- Die Partei $P_1$ („Zentralhub“) empfängt die voraggregierte Informationsvektoren und -matrizen von $P_2, P_3$, aggregiert diese mit ihren eigenen und sendet die aggregierte Daten an den Agenten $P_{N+1}$.

**Protokoll 4.1 (Sicheres verteiltes Informationsfilter $\pi_{\mathrm{IF}}$)**

- **Parameter**: Alle Parteien kennen (KeyGen, Enc, Dec, Eval), die Netzwerkgröße $N$, die Rundenzahl $K$ und den Sicherheitsparameter $n$.
- **Eingaben**:
  - Jede Partei $P_j$ mit $j \in \{1, \ldots, N\}$ (Sensor) hat eine Sequenz $\hat{\underline{i}}_1^j, \mathbf{I}_1^j, \hat{\underline{i}}_2^j, \mathbf{I}_2^j$, ..., $\hat{\underline{i}}_K^j, \mathbf{I}_K^j$, einen Wert $\mathcal{H}_j = P_{MyHub(j)}$ und eine Indexmenge $\mathcal{N}_j = MySenders(j)$ nach der Def. 4.2.
  - Die Partei $P_{N+1}$ (Agent) hat keine bzw. eine leere Eingabe $\lambda$.
- **Das Protokoll**:
  1. $P_{N+1}$ berechnet $(pk, sk) \leftarrow \mathsf{KeyGen}\,(1^n)$.
  2. $P_{N+1}$ sendet $pk$ an $P_1, \ldots, P_N$.
  3. Alle Parteien setzen eine interne Variable $k \coloneqq 1$.
  4. Jede $P_{j\in\{1,\ldots,N\}}$ berechnet $[\![\hat{\underline{i}}_k^j]\!] \leftarrow \mathsf{Enc}(pk, \hat{\underline{i}}_k^j), [\![\mathbf{I}_k^j]\!] \leftarrow \mathsf{Enc}(pk, \mathbf{I}_k^j)$.
  5. Jede $P_{j\in\{4,\ldots,N\}}$ sendet $[\![\hat{\underline{i}}_k^j]\!], [\![\mathbf{I}_k^j]\!]$ an ihren jeweiligen Hub $\mathcal{H}_j$.
  6. Jede $P_{j\in\{2,3\}}$ berechnet elementweise für alle $s_1, s_2, \ldots \in \mathcal{N}_j$:
  $$[\![\hat{\underline{i}}_k^{j+}]\!] \leftarrow [\![\hat{\underline{i}}_k^j]\!] +_{pk} [\![\hat{\underline{i}}_k^{s_1}]\!] +_{pk} [\![\hat{\underline{i}}_k^{s_2}]\!] +_{pk} \ldots$$
  $$[\![\mathbf{I}_k^{j+}]\!] \leftarrow [\![\mathbf{I}_k^j]\!] +_{pk} [\![\mathbf{I}_k^{s_1}]\!] +_{pk} [\![\mathbf{I}_k^{s_2}]\!] +_{pk} \ldots$$
  7. Jede $P_{j\in\{2,3\}}$ sendet $[\![\hat{\underline{i}}_k^{j+}]\!], [\![\mathbf{I}_k^{j+}]\!]$ an $\mathcal{H}_j = P_1$.
  8. $P_1$ berechnet $[\![\hat{\underline{i}}_k]\!] \leftarrow [\![\hat{\underline{i}}_k^1]\!] +_{pk} [\![\hat{\underline{i}}_k^{2+}]\!] +_{pk} [\![\hat{\underline{i}}_k^{3+}]\!], [\![\mathbf{I}_k]\!] \leftarrow [\![\mathbf{I}_k^1]\!] +_{pk} [\![\mathbf{I}_k^{2+}]\!] +_{pk} [\![\mathbf{I}_k^{3+}]\!]$.
  9. $P_1$ sendet $[\![\hat{\underline{i}}_k]\!], [\![\mathbf{I}_k]\!]$ an $P_{N+1}$.
  10. $P_{N+1}$ berechnet $\hat{\underline{i}}_k \coloneqq \mathsf{Dec}(sk, [\![\hat{\underline{i}}_k]\!]), \mathbf{I}_k \coloneqq \mathsf{Dec}(sk, [\![\mathbf{I}_k]\!])$ und gibt sie aus.
  11. Alle Parteien setzen $k \coloneqq k+1$ und, wenn $k \leq K$, springen zum Schritt 4 zurück.

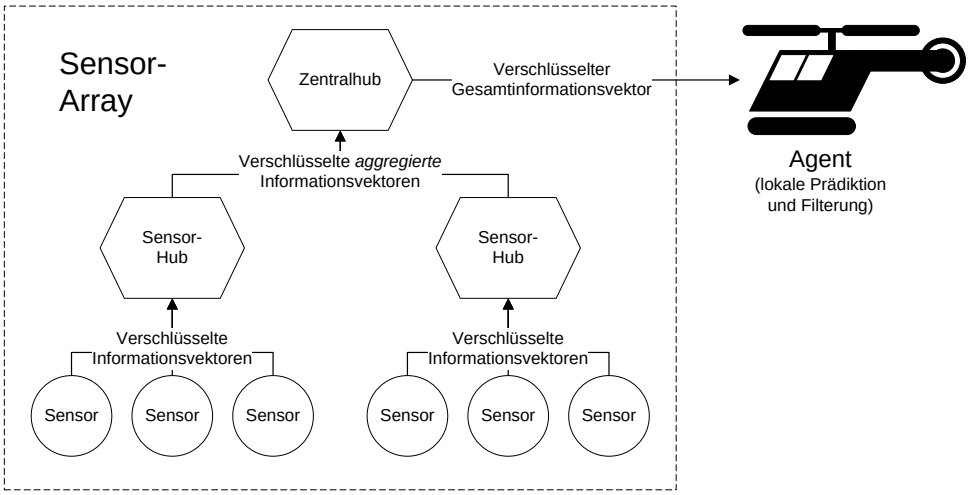


**Abbildung 4.1:** Schematischer Aufbau des Protokolls 4.1 ($\pi_{\mathrm{IF}}$) mit zwei Zwischenhubs und 6 weiteren Sensoren (beispielhaft).

- Die Parteien $P_2, P_3$ („Hubs“) empfangen die rohen (unaggegrierten) Daten von den den Sensoren $P_4, \ldots, P_N$, aggregieren diese mit ihren eigenen und senden sie an $P_1$ weiter.
- Jeder Sensor $P_j$ mit $j \in \{4, \ldots, N\}$ sendet seine Messungen an $P_2$, wenn $j \leq \lceil \frac{N}{2} \rceil$, bzw. an $P_3$, wenn $j > \lceil \frac{N}{2} \rceil$.

## 4.3 Sicherheitsbeweis

Wir wollen die Sicherheit des Protokolls 4.1 gegenüber statischen semiehrlichen PPT-beschränkten Angreifern beweisen. Weil $f_{\mathrm{IF}}$ eine deterministische Funktionalität ist, genügt es für unseren konkreten Sicherheitsbeweis, die vereinfachte Formulierung der Sicherheitsdefinition 2.8 zu erfüllen, nämlich, dass unser Protokoll die gewünschte Funktionalität sowohl korrekt, als auch vertraulich umsetzt.

### 4.3.1 Korrektheit

***Satz 4.1*** *Unter der Annahme, dass* (KeyGen, Enc, Dec, Eval) *ein asymmetrisches Verschlüsselungsschema mit additivem Homomorphismus und perfekter Korrektheit ist, implementiert das Protokoll $\pi_{\mathrm{IF}}$ die Funktionalität $f_{\mathrm{IF}}$ korrekt, d. h. für alle $k \in \{1, \ldots, K\}$ gilt:*

$$\hat{\underline{i}}_k = \sum_{j=1}^{N} \hat{\underline{i}}_k^j \quad \textit{und} \quad \mathbf{I}_k = \sum_{j=1}^{N} \mathbf{I}_k^j.$$

Beweis. Wir betrachten zuerst den Term $\hat{\underline{i}}_1$, d. h. die Ausgabe von $P_{N+1}$ in der Kommunikationsrunde $k = 1$. Aus der perfekten Korrektheit des homomorphen Verschlüsselungsschemas nach der Def. 2.4 folgt:

$$\begin{aligned}\hat{\underline{i}}_1 &= \mathsf{Dec}(sk, [\![\hat{\underline{i}}_1]\!]) = \mathsf{Dec}(sk, [\![\hat{\underline{i}}_1^1]\!] +_{pk} [\![\hat{\underline{i}}_1^{2+}]\!] +_{pk} [\![\hat{\underline{i}}_1^{3+}]\!]) \\ &= \mathsf{Dec}(sk, [\![\hat{\underline{i}}_1^1]\!]) + \mathsf{Dec}(sk, [\![\hat{\underline{i}}_1^{2+}]\!]) + \mathsf{Dec}(sk, [\![\hat{\underline{i}}_1^{3+}]\!]) \\ &= \hat{\underline{i}}_1^1 + \mathsf{Dec}(sk, [\![\hat{\underline{i}}_1^{2+}]\!]) + \mathsf{Dec}(sk, [\![\hat{\underline{i}}_1^{3+}]\!]). \end{aligned} \tag{4.2}$$

Analog können wir aus der Protokolldefinition ableiten, dass

$$\begin{aligned}\mathsf{Dec}(sk, [\![\hat{\underline{i}}_1^{2+}]\!]) &= \mathsf{Dec}(sk, [\![\hat{\underline{i}}_1^2]\!] +_{pk} [\![\hat{\underline{i}}_1^4]\!] +_{pk} [\![\hat{\underline{i}}_1^5]\!] +_{pk} \ldots +_{pk} [\![\hat{\underline{i}}_1^H]\!]) \\ &= \hat{\underline{i}}_1^2 + \hat{\underline{i}}_1^4 + \ldots + \hat{\underline{i}}_1^H = \hat{\underline{i}}_1^2 + \sum_{j=4}^{H} \hat{\underline{i}}_1^j \\ \mathsf{Dec}(sk, [\![\hat{\underline{i}}_1^{3+}]\!]) &= \mathsf{Dec}(sk, [\![\hat{\underline{i}}_1^3]\!] +_{pk} [\![\hat{\underline{i}}_1^{H+1}]\!] +_{pk} [\![\hat{\underline{i}}_1^{H+2}]\!] +_{pk} \ldots +_{pk} [\![\hat{\underline{i}}_1^N]\!])) \\ &= \hat{\underline{i}}_1^3 + \hat{\underline{i}}_1^{\lfloor \frac{N}{2}+1 \rfloor} + \ldots + \hat{\underline{i}}_1^N = \hat{\underline{i}}_1^3 + \sum_{j=H+1}^{N} \hat{\underline{i}}_1^j. \end{aligned}$$

Setzen wir die beiden Gleichungen wieder in die (4.2) ein, erhalten wir

$$\hat{\underline{i}}_1 = \hat{\underline{i}}_1^1 + \hat{\underline{i}}_1^2 + \sum_{j=4}^{H} \hat{\underline{i}}_1^j + \hat{\underline{i}}_1^3 + \sum_{j=H+1}^{N} \hat{\underline{i}}_1^j = \hat{\underline{i}}_1^1 + \hat{\underline{i}}_1^2 + \hat{\underline{i}}_1^3 + \sum_{j=4}^{H} \hat{\underline{i}}_1^j + \sum_{j=H+1}^{N} \hat{\underline{i}}_1^j = \sum_{j=1}^{N} \hat{\underline{i}}_1^j.$$

Damit haben wir die Korrektheit der Berechnung von $\hat{\underline{i}}_1$ gezeigt. Die Korrektheitsbeweise für $\hat{\underline{i}}_2, \ldots, \hat{\underline{i}}_K$ und für $\mathbf{I}_1, \ldots, \mathbf{I}_K$ verlaufen analog, wodurch wir letztendlich die Korrektheit des Gesamtprotokolls nachweisen können. □

### 4.3.2 Vertraulichkeit

Bevor wir zu der Analyse der Vertraulichkeitsgarantien des Protokolls übergehen, muss eine weitere Annahme über die Korruptionsstrategie des Angreifers getroffen werden. Im Gegensatz zu den klassischen Zweiparteien-Protokollen aus der Kryptographieliteratur [41, 52] sind am $\pi_{\text{IF}}$ viel mehr Parteien beteiligt, und ein vollständiger Sicherheitsbeweis muss nicht nur die Korruption einzelner Parteien, sondern auch jede Kombination von Parteien betrachten, die gleichzeitig korrumpiert werden könnten.

Dies wird vor allem gefährlich, wenn die schlüsselerzeugende Partei $P_{N+1}$ (Agent) gleichzeitig mit einer oder mehreren aggregierenden Parteien $P_1, P_2, P_3$ (Hubs) korrumpiert wird, weil der Angreifer in diesem Fall den $sk$ vom Agent benutzen kann, um die einzelnen Nachrichten zu entschlüsseln, die die ehrlichen Sensoren an die

Hubs schicken, und damit ihre Eingaben zu lernen. Deswegen benötigen wir eine zusätzliche Annahme, dass der Angreifer nicht gleichzeitig eine schlüsselerzeugende und eine aggregierende Partei korrumpieren kann. Diese ist gut vertretbar, weil eine gleichzeitige Korruption eines Agenten und eines zentralen Sensornetzwerkknoten grundsätzlich auch auf nicht-kryptographische Weise verhindert werden kann.

***Satz 4.2*** *Unter den Annahmen, dass a)* (KeyGen, Enc, Dec, Eval) *ein IND-CPA sicheres asymmetrisches Verschlüsselungsschema mit komplexitätstheoretisch starkem additivem Homomorphismus ist, und dass b) kein Angreifer $P_{N+1}$ gleichzeitig mit $P_1, P_2$ oder $P_3$ korrumpieren kann, berechnet $\pi_{\mathrm{IF}}$ die Funktionalität $f_{\mathrm{IF}}$ in der Anwesenheit der statischen semiehrlichen PPT-Angreifern sicher.*

Beweis. Zuerst betrachten wir die Fälle der Korruption der einzelnen Parteien.

**Korruption einzelner Sensoren** Im einfachsten Fall wird einer der Nicht-Hub-Sensoren $P_{j\in\{4,\ldots,N\}}$ korrumpiert. Zuerst betrachten wir den Korruptionsfall von $P_4$ und konstruieren einen Simulator $\mathcal{S}_4$, der die Nachrichten, die $P_4$ in einer realen Protokollausführung erhält, so simuliert, dass seine Ausgabe komplexitätstheoretisch nicht von der realen View von $P_4$ unterscheidbar ist:

$$\{\mathcal{S}_4(1^n, E_4, \lambda)\} \stackrel{\mathrm{c}}{\equiv} \{\mathrm{view}_4^{\pi_{\mathrm{IF}}}(E_1, E_2, \ldots, E_N, \lambda)\}\,,$$

wobei die Eingaben $E_j$ wie in (4.1) definiert sind.

Die Eingabe von $\mathcal{S}_4$ besteht aus der Ein- und Ausgabe von $P_4$, sowie dem Sicherheitsparameter $n$, also explizit ausgedrückt $(1^n, \hat{\underline{\imath}}_1^4, \mathbf{I}_1^4, \ldots, \hat{\underline{\imath}}_K^4, \mathbf{I}_K^4)$. Die $\mathrm{view}_4^{\pi_{\mathrm{IF}}}$ besteht aus den Eingaben von $P_4$, seinem Zufallsband $r^4$ und der einzigen Nachricht, die $P_4$ im ganzen Protokollverlauf bekommt: $pk$ von $P_{N+1}$.

$$\mathrm{view}_4^{\pi_{\mathrm{IF}}}(E_1, E_2, \ldots, E_N, \lambda) = (\hat{\underline{\imath}}_1^4, \mathbf{I}_1^4, \ldots, \hat{\underline{\imath}}_K^4, \mathbf{I}_K^4, r^4; pk)$$

Damit lässt sich nun $\mathcal{S}_4$ trivial konstruieren und geht wie folgt vor:

1. $\mathcal{S}_4$ zieht für $P_4$ ein uniform verteiltes Zufallsband $r^4$ der ausreichenden Länge.
2. $\mathcal{S}_4$ berechnet $(pk, sk) \leftarrow \mathsf{KeyGen}(1^n)$ mit unabhängigem Zufall.
3. $\mathcal{S}_4$ gibt aus: $(\hat{\underline{\imath}}_1^4, \mathbf{I}_1^4, \ldots, \hat{\underline{\imath}}_K^4, \mathbf{I}_K^4, r^4; pk)$.

Dass die Ausgabe des Simulators nicht von der realen View der $P_4$ unterscheidbar ist, folgt unmittelbar daraus, dass die Eingaben in beiden Fällen identisch sind, die Zufallsbänder beide gleichverteilt sind, und der $pk$ vom Simulator genauso erzeugt wird, wie es eine ehrliche $P_{N+1}$ tun würde, und dadurch in der Simulation identisch wie im Realen verteilt ist. Die Simulatoren $\mathcal{S}_5, \ldots, \mathcal{S}_N$ gehen analog vor, um ebenfalls ununterscheidbare Views für $P_5, \ldots, P_N$ zu erzeugen.

**Korruption einzelner Zwischenhubs** Nun betrachten wir den Fall der Einzelkorruption der Hubs $P_2, P_3$ und fokussieren uns zuerst auf $P_2$. Dieser hat analog $P_4$ die Eingabe $E_4 = (\hat{\underline{i}}_1^2, \mathbf{I}_1^2, \ldots, \hat{\underline{i}}_K^2, \mathbf{I}_K^2)$ und keine Ausgabe, bekommt aber neben $pk$ insgesamt $K\,|\mathcal{N}_2|$ weitere Nachrichtenpaare, nämlich die verschlüsselten Informationsvektoren und -matrizen von $P_{j\in\mathcal{N}_2}$, wobei $K$ die Zeitrundenanzahl und $\mathcal{N}_2 = \{4, \ldots, H\}$ die Menge der Sensoren ist, die an $P_2$ ihre Messungen senden. Damit gilt für seine View:

$$\text{view}_2^{\pi_{\text{IF}}}(E_1, E_2, \ldots, E_N, \lambda) = (E_2, r^2; pk, [\![\hat{\underline{i}}_1^4]\!], [\![\mathbf{I}_1^4]\!], \ldots, [\![\hat{\underline{i}}_K^H]\!], [\![\mathbf{I}_K^H]\!]). \tag{4.3}$$

Diese Nachrichten muss der Simulator $\mathcal{S}_2$ zusätzlich zu dem trivial simulierbaren $pk$ erzeugen und geht dazu wie folgt vor:

1. $\mathcal{S}_2$ zieht für $P_2$ ein uniform verteiltes Zufallsband $r^2$ der ausreichenden Länge.
2. $\mathcal{S}_2$ berechnet $(pk, sk) \leftarrow \mathsf{KeyGen}(1^n)$ mit unabhängigem Zufall.
3. $\mathcal{S}_2$ führt $\mathsf{Enc}(pk, \underline{0})$ und $\mathsf{Enc}(pk, \mathbf{0})$ jeweils $K\,|\mathcal{N}_2|$ mal mit unabhängigem Zufall aus (wobei $\underline{0}$ und $\mathbf{0}$ hier die gleichen Dimensionen wie $\hat{\underline{i}}_k$ und $\mathbf{I}_k$ haben), um Chiffrat-Arrays $[\![\underline{0}]\!], [\![\mathbf{0}]\!], \ldots, [\![\underline{0}]\!], [\![\mathbf{0}]\!]$ zu erhalten.
4. $\mathcal{S}_2$ gibt aus: $(\hat{\underline{i}}_1^2, \mathbf{I}_1^2, \ldots, \hat{\underline{i}}_K^2, \mathbf{I}_K^2, r^2; pk, [\![\underline{0}]\!], [\![\mathbf{0}]\!], \ldots, [\![\underline{0}]\!], [\![\mathbf{0}]\!])$.

Offensichtlich kann der Simulator die Eingaben (Messungen) der ehrlichen Parteien (Sensoren), deren Nachrichten er simuliert, nicht wissen, deshalb muss er die letzteren entweder raten oder blind fälschen. Weil die Nachrichten aber, per Annahme, mit einem IND-CPA-sicheren Verschlüsselungsschema verschlüsselt werden, ist es tatsächlich irrelevant, welche Klartexte zum Fälschen verwendet werden: ein „Nullchiffrat“ (Verschlüsselung von einer Nachricht, die nur aus Null-Bits besteht) ist von jedem anderen Chiffrat komplexitätstheoretisch ununterscheidbar. Dabei spielt auch keine Rolle, ob eine Zahl oder ein ganzes Array (Nullvektor bzw. -matrix) verschlüsselt werden, denn laut [48, Satz 11.6], wenn CPA-Sicherheit für ein Chiffrat gewährt ist, ist sie auch für mehrere Chiffrate gegeben, solange die Gesamtlänge der Klartexte gleich ist, was im Schritt 3 explizit beachtet wird.

Daraus und aus der Gleichung (4.3) leiten wir das folgende *Hybridargument* ab:

$$\begin{aligned}
\{(E_2, r^2; pk, [\![\hat{\underline{i}}_1^4]\!], [\![\mathbf{I}_1^4]\!], \ldots, [\![\hat{\underline{i}}_K^H]\!], [\![\mathbf{I}_K^H]\!])\} &\overset{c}{\equiv} \{(E_2, r^2; pk, [\![\underline{0}]\!], [\![\mathbf{I}_1^4]\!], \ldots, [\![\hat{\underline{i}}_K^H]\!], [\![\mathbf{I}_K^H]\!])\} \\
&\overset{c}{\equiv} \{(E_2, r^2; pk, [\![\underline{0}]\!], [\![\mathbf{0}]\!], \ldots, [\![\hat{\underline{i}}_K^H]\!], [\![\mathbf{I}_K^H]\!])\} \\
&\vdots \\
&\overset{c}{\equiv} \{(E_2, r^2; pk, [\![\underline{0}]\!], [\![\mathbf{0}]\!], \ldots, [\![\underline{0}]\!], [\![\mathbf{I}_K^H]\!])\} \\
&\overset{c}{\equiv} \{(E_2, r^2; pk, [\![\underline{0}]\!], [\![\mathbf{0}]\!], \ldots, [\![\underline{0}]\!], [\![\mathbf{0}]\!])\} \\
&\equiv \{\mathcal{S}_2(1^n, E_2, \lambda)\}.
\end{aligned}$$

Intuitiv bedeutet das Argument, dass wenn wir die Verteilung der realen View von $P_2$ so manipulieren, dass die erste Nachricht von $P_4$ darin durch ein Nullvektorchiffrat ersetzt wird, die originale und die manipulierte Verteilung komplexitätstheoretisch ununterscheidbar bleiben. Denn wenn ein PPT-Algorithmus sie an der einen manipulierten Stelle unterscheiden könnte, würde er damit ein reales Chiffrat von einem Nullchiffrat unterscheiden und die IND-CPA-Sicherheit der Verschlüsselung brechen. Wenn wir nun die zweite Nachricht von $P_4$, die erste von $P_5$, usw. analog austauschen, bleiben die Verteilungen immer schrittweise ununterscheidbar, bis wir alle echten Nachrichten durch Nullchiffrate ersetzt haben – und das ist ja gerade die Ausgabe von $\mathcal{S}_2$. Und weil die Ununterscheidbarkeitsrelation transitiv ist, haben wir somit gezeigt, dass die Simulatorausgabe komplexitätstheoretisch nicht von realen View unterscheidbar ist. Der Simulator $\mathcal{S}_3$ geht analog vor.

**Korruption des zentralen Hubs** Der Zentralhub $P_1$ erhält ähnlich wie die anderen beiden Hubs neben dem $pk$ weitere Nachrichten, diese sind aber keine frischen Chiffrate, sondern Ergebnisse von $\mathsf{Eval}$-Aufrufe.

$$\mathrm{view}_1^{\pi_{\mathrm{IF}}}(E_1, E_2, \ldots, E_N, \lambda) = (E_1, r^1; pk, [\![\hat{\underline{i}}_1^{2+}]\!], [\![\mathbf{I}_1^{2+}]\!], \ldots, [\![\hat{\underline{i}}_K^{3+}]\!], [\![\mathbf{I}_K^{3+}]\!])$$

Trotzdem ändert sich am Simulatorvorgehen grundsätzlich nichts:

1. $\mathcal{S}_1$ zieht für $P_1$ ein uniform verteiltes Zufallsband $r^1$ der ausreichenden Länge.
2. $\mathcal{S}_1$ berechnet $(pk, sk) \leftarrow \mathsf{KeyGen}(1^n)$ mit unabhängigem Zufall.
3. $\mathcal{S}_1$ führt $\mathsf{Enc}(pk, \underline{0})$ und $\mathsf{Enc}(pk, \mathbf{0})$ $2K$ mal mit jeweils unabhängigem Zufall aus, um die Chiffrat-Arrays $[\![\underline{0}]\!], [\![\mathbf{0}]\!], \ldots, [\![\underline{0}]\!], [\![\mathbf{0}]\!]$ zu erhalten.
4. $\mathcal{S}_1$ gibt aus: $(\hat{\underline{i}}_1^1, \mathbf{I}_1^1, \ldots, \hat{\underline{i}}_K^1, \mathbf{I}_K^1, r^1; pk, [\![\underline{0}]\!], [\![\mathbf{0}]\!], \ldots, [\![\underline{0}]\!], [\![\mathbf{0}]\!])$.

Auch hier ist ein Hybridargument anwendbar, allerdings ist es etwas komplexer, denn

$$\begin{aligned}
[\![\hat{\underline{i}}_1^{2+}]\!] &= [\![\hat{\underline{i}}_1^{2}]\!] +_{pk} [\![\hat{\underline{i}}_1^{4}]\!] +_{pk} \ldots +_{pk} [\![\hat{\underline{i}}_1^{H}]\!] \\
[\![\mathbf{I}_1^{2+}]\!] &= [\![\mathbf{I}_1^{2}]\!] +_{pk} [\![\mathbf{I}_1^{4}]\!] +_{pk} \ldots +_{pk} [\![\mathbf{I}_1^{H}]\!] \\
&\vdots \\
[\![\hat{\underline{i}}_K^{3+}]\!] &= [\![\hat{\underline{i}}_K^{3}]\!] +_{pk} [\![\hat{\underline{i}}_K^{H+1}]\!] +_{pk} \ldots +_{pk} [\![\hat{\underline{i}}_K^{N}]\!] \\
[\![\mathbf{I}_K^{3+}]\!] &= [\![\mathbf{I}_K^{3}]\!] +_{pk} [\![\mathbf{I}_K^{H+1}]\!] +_{pk} \ldots +_{pk} [\![\mathbf{I}_K^{N}]\!].
\end{aligned}$$

Per Annahme a) im Satz 4.2 hat unser Verschlüsselungsschema *komplexitätstheoretisch starken* Homomorphismus. Aus der Def. 2.5 folgt, dass

$$\left\{ [\![\hat{\underline{i}}_1^{2}]\!] +_{pk} [\![\hat{\underline{i}}_1^{4}]\!] +_{pk} \ldots +_{pk} [\![\hat{\underline{i}}_1^{H}]\!] \right\} \overset{\mathrm{c}}{\equiv} \left\{ [\![\hat{\underline{i}}_1^{2} + \hat{\underline{i}}_1^{4} + \ldots + \hat{\underline{i}}_1^{H}]\!]_{pk} \right\},$$

und aus der IND-CPA-Sicherheit des Schemas folgt außerdem, dass

$$\left\{[\![\hat{\underline{i}}_1^2 + \hat{\underline{i}}_1^4 + \ldots + \hat{\underline{i}}_1^H]\!]_{pk}\right\} \overset{\text{c}}{\equiv} \left\{[\![\underline{0}]\!]_{pk}\right\}.$$

In anderen Worten, ist ein Eval-Ergebnis unter der Annahme a) komplexitätstheoretisch nicht von einem frischen Nullchiffrat unterscheidbar. Damit lässt sich für $\mathcal{S}_1$ ein Hybridargument analog wie für $\mathcal{S}_2$ aufstellen, um letztendlich zu zeigen, dass die Gesamtausgabe von $\mathcal{S}_1$ nicht von der realen View von $P_1$ unterscheibar ist.

**Korruption des Agenten** Der Agent $P_{N+1}$ hat im Gegensatz zu den Sensoren keine Eingaben, sondern nur Ausgaben, nämlich $\hat{\underline{i}}_1, \mathbf{I}_1, \ldots, \hat{\underline{i}}_K, \mathbf{I}_K$, die der Simulator $\mathcal{S}_{N+1}$ als Eingabe bekommt. $P_{N+1}$ erhält $2K$ Nachrichten von $P_1$, die aber alle Chiffrate seiner Ausgaben sind.

$$\text{view}_{N+1}^{\pi_{\text{IF}}}(E_1, E_2, \ldots, E_N, \lambda) = (r^{N+1}; pk, [\![\hat{\underline{i}}_1]\!], [\![\mathbf{I}_1]\!], \ldots, [\![\hat{\underline{i}}_K]\!], [\![\mathbf{I}_K]\!])$$

Deshalb geht der Simulator einfach wie folgt vor:

1. $\mathcal{S}_{N+1}$ zieht für $P_{N+1}$ ein uniform verteiltes Zufallsband $r^{N+1}$ der ausreichenden Länge, um $\mathsf{KeyGen}(1^n)$ auszuführen.
2. $\mathcal{S}_{N+1}$ berechnet $(pk, sk) \leftarrow \mathsf{KeyGen}(1^n)$ mit $r^{N+1}$.
3. $\mathcal{S}_{N+1}$ führt $\mathsf{Enc}_{pk}(\hat{\underline{i}}_1), \mathsf{Enc}(pk, \mathbf{I}_1), \ldots, \mathsf{Enc}_{pk}(\hat{\underline{i}}_K), \mathsf{Enc}(pk, \mathbf{I}_K)$ mit unabhängigem Zufall aus, um die Chiffrat-Arrays $[\![\hat{\underline{i}}_1]\!], [\![\mathbf{I}_1]\!], \ldots, [\![\hat{\underline{i}}_K]\!], [\![\mathbf{I}_K]\!]$ zu erhalten.
4. $\mathcal{S}_{N+1}$ gibt aus: $(r^{N+1}; [\![\hat{\underline{i}}_1]\!], [\![\mathbf{I}_1]\!], \ldots, [\![\hat{\underline{i}}_K]\!], [\![\mathbf{I}_K]\!])$.

Weil die Ausgabe des Simulators sowie die reale View das Zufallsband enthalten, aus dem das Schlüsselpaar erzeugt wird, ist es wichtig, dass im Schritt 2 $r^{N+1}$ zur Schlüsselerzeugung verwendet wird. Außerdem folgt daraus, dass jeder Unterscheider die im Schritt 3 gefälschten Nachrichten wieder effizient entschlüsseln kann: dies ist aber ungefährlich, weil der Simulator die korrekten Klartexte für alle Nachrichten als Eingabe bekommt, und, wie wir bereits vorher gesehen hatten, ein frisches Chiffrat von einem Eval-Ergebnis komplexitätstheoretisch ununterscheidbar ist. Damit lässt sich wieder analog ein Hybridargument aufstellen, um zu zeigen, dass die Ausgabe von $\mathcal{S}_{N+1}$ nicht von der realen View von $P_{N+1}$ unterscheidbar ist.

**Mehrfachkorruption** Nun betrachten wir die unterschiedlichen Szenarien, in denen mehrere Parteien von einem Angreifer gleichzeitig korrumpiert werden. Dabei beachten wir per Annahme b) nicht den Fall, in dem der Agent und ein oder mehrere Hubs korrumpiert werden. Es bleiben drei Szenarien zu untersuchen:

- Eine Untermenge der Sensoren ohne Hubs ist korrumpiert;
- Eine Untermenge der Sensoren ohne Hubs und der Agent sind korrumpiert;
- Eine Untermenge der Sensoren *und* eine Untermenge der Hubs sind korrumpiert.

**Korruption mehrerer Sensoren** Der einfachste Fall der Mehrfachkorruption ist, dass ein Angreifer eine Untermenge der Sensoren korrumpiert, in der sich keine Hubs befinden. Wir bezeichnen die Menge der Indizes der korrumpierten Sensoren mit $Bad \subseteq \{4, \ldots, N\}$, und die korrumpierten Sensoren selbst mit $\{P_j : j \in Bad\}$ bzw. abgekürzt $\{P_{j \in Bad}\}$. Gesucht ist ein Simulator $\mathcal{S}_{\{j:j \in Bad\}}$, abgekürzt $\mathcal{S}_{Bad}$, dessen Ausgabe nicht von der *gesamten* Verbundview aller $\{P_{j \in Bad}\}$ unterscheidbar ist.

Dies hat zufolge, dass der öffentliche Schlüssel $pk$ in jeder simulierten View gleich sein muss, sonst erkennt jeder Unterscheider trivial, welche Verbundview echt und welche gefälscht wurde. Aus diesem Grund kann $\mathcal{S}_{Bad}$ keine Simulatoren für einzelne Sensoren (z. B. $\mathcal{S}_4$), wiederverwenden, denn sie würden jeweils einen eigenen $pk$ ausgeben. Stattdessen geht er wie folgt vor:

1. $\mathcal{S}_{Bad}$ zieht für jede $P_{j \in Bad}$ ein uniform verteiltes Zufallsband $r^j$ der ausreichenden Länge.
2. $\mathcal{S}_{N+1}$ berechnet $(pk, sk) \leftarrow \mathsf{KeyGen}(1^n)$ mit unabhängigem Zufall.
3. $\mathcal{S}_{Bad}$ gibt für jedes $j \in Bad$ aus: $(\hat{\underline{\iota}}_1^j, \mathbf{I}_1^j, \ldots, \hat{\underline{\iota}}_K^j, \mathbf{I}_K^j, r^j; pk)$.

Dass die Ausgabe von $\mathcal{S}_{Bad}$ identisch wie die realen Views der korrumpierten Parteien $P_{j \in Bad}$ verteilt ist, ist naheliegend: der Simulator kennt alle ihre Eingaben, die Zufallsbänder werden identisch gezogen, und die einzige Nachricht, die sie bekommen, ist $pk$. Dieser wird genauso erzeugt, wie es eine ehrliche $P_{N+1}$ tun würde, und ist, wie auch im echten Protokoll, in allen simulierten Views gleich.

**Korruption mehrerer Sensoren und des Agenten** Wir erweitern die Korruptionsstragie des Angreifers vom letzten Abschnitt um den Agent selbst, bezeichnen aber mit $Bad$ weiterhin nur die Menge der korrumpierten Sensorindizes, ohne den Agenten. Dabei beobachten wir, dass $\{P_{j \in Bad}\}$ in keinem Schritt des Protokolls an den Agent $P_{N+1}$ Nachrichten senden, aber der Agent sendet seinen $pk$ an alle. Dies erlaubt uns, den $\mathcal{S}_{N+1}$ im $\mathcal{S}_{Bad \cup \{N+1\}}$ wiederzuverwenden, und zwar wie folgt:

1. $\mathcal{S}_{Bad \cup \{N+1\}}$ zieht für jede $P_{j \in Bad}$ ein uniform verteiltes Zufallsband $r^j$ der ausreichenden Länge.
2. $\mathcal{S}_{Bad \cup \{N+1\}}$ ruft $\mathcal{S}_{N+1}$ als Unterfunktion auf und gibt seine Ausgabe selbst aus.

3. $\mathcal{S}_{Bad\cup\{N+1\}}$ gibt für jedes $j \in Bad$ aus: $(\hat{\underline{i}}_1^j, \mathbf{I}_1^j, \ldots, \hat{\underline{i}}_K^j, \mathbf{I}_K^j, r^j; pk)$, wobei $pk$ hier aus der Ausgabe von $\mathcal{S}_{N+1}$ übernommen wird.

Aus den bisherigen Überlegungen liegt es nahe, dass die Ausgabe von $\mathcal{S}_{Bad\cup\{N+1\}}$ nicht von der realen Verbundview unterscheidbar sein kann.

**Korruption mehrerer Sensoren und Hubs** Wir bezeichnen weiterhin die Menge der korrumpierten Sensorindizes ohne Hubs mit $Bad$, und die Menge der korrumpierten Hubs nun mit $BadHub \subseteq \{1, 2, 3\}$. In diesem Szenario müssen wir beachten, dass die korrumpierten Parteien auch unter sich kommunizieren und der Simulator $\mathcal{S}_{Bad\cup BadHub}$ diese „interne“ Kommunikation in sich konsistent fälschen muss, indem er sowohl die Eingaben, als auch die von ihm selbst erzeugten Zufallsbänder der korrumpierten Parteien miteinbezieht:

1. $\mathcal{S}_{Bad\cup BadHub}$ ruft $\mathcal{S}_{Bad}$ als Unterfunktion auf und gibt seine Ausgabe aus.
2. $\mathcal{S}_{Bad\cup BadHub}$ zieht für jede $P_{j\in BadHub}$ ein uniform verteiltes Zufallsband $r^{j\in BadHub}$ der ausreichenden Länge.
3. Wenn $2 \in BadHub$:
   a) $\mathcal{S}_{Bad\cup BadHub}$ berechnet für jedes $j \in \mathcal{N}_2 \cap Bad$ und alle $k \in \{1, \ldots, K\}$ $\underline{u}_k^j \leftarrow \mathsf{Enc}(pk, \hat{\underline{i}}_k^j)$ und $\mathbf{U}_k^j \leftarrow \mathsf{Enc}(pk, \mathbf{I}_k^j)$ mit dem öffentlichen Schlüssel $pk$ und dem Zufallsband $r^j$ aus der Ausgabe von $\mathcal{S}_{Bad}$.
   b) $\mathcal{S}_{Bad\cup BadHub}$ berechnet für jedes $j' \in \mathcal{N}_2 \setminus Bad$ und alle $k \in \{1, \ldots, K\}$ $\underline{u}_k^{j'} \leftarrow \mathsf{Enc}(pk, \underline{0})$ und $\mathbf{U}_k^{j'} \leftarrow \mathsf{Enc}(pk, \mathbf{0})$ mit unabhängigem Zufall.
   c) $\mathcal{S}_{Bad\cup BadHub}$ gibt aus: $(\hat{\underline{i}}_1^2, \mathbf{I}_1^2, \ldots, \hat{\underline{i}}_K^2, \mathbf{I}_K^2, r^2; pk, \underline{u}_1^4, \mathbf{U}_1^4, \ldots, \underline{u}_K^H, \mathbf{U}_K^H)$.
4. Wenn $3 \in BadHub$, geht $\mathcal{S}_{Bad\cup BadHub}$ analog für $j \in \mathcal{N}_3 \cap Bad$ bzw. $j' \in \mathcal{N}_3 \setminus Bad$ vor.
5. Wenn $1 \in BadHub$:
   a) Wenn $2 \in BadHub$, berechnet $\mathcal{S}_{Bad\cup BadHub}$ für $k \in \{1, \ldots, K\}$

$$\underline{v}_k^{2+} \leftarrow \mathsf{Enc}(pk, \hat{\underline{i}}_2^k) +_{pk} \underline{u}_k^4 +_{pk} \ldots +_{pk} \underline{u}_k^H \quad \text{und}$$
$$\mathbf{V}_k^{2+} \leftarrow \mathsf{Enc}(pk, \mathbf{I}_2^k) +_{pk} \mathbf{U}_k^4 +_{pk} \ldots +_{pk} \mathbf{U}_k^H,$$

   ansonsten

$$\underline{v}_k^{2+} \leftarrow \mathsf{Enc}(pk, \underline{0}) \quad \text{und} \quad \mathbf{V}_k^{2+} \leftarrow \mathsf{Enc}(pk, \mathbf{0}).$$

   b) $\mathcal{S}_{Bad\cup BadHub}$ berechnet analog $\underline{v}_k^{3+}, \mathbf{V}_k^{3+}$, um die Nachrichten von $P_3$ an $P_1$ zu simulieren.

c) $\mathcal{S}_{Bad \cup BadHub}$ gibt aus: $(\hat{\underline{i}}_1^1, \mathbf{I}_1^1, \ldots, \hat{\underline{i}}_K^1, \mathbf{I}_K^1, r^1; pk, \underline{v}_1^{2+}, \mathbf{V}_1^{2+}, \ldots, \underline{v}_K^{3+}, \mathbf{V}_K^{3+})$.

Dass die simulierten Views der Parteien $\{P_{j \in Bad}\}$ von den realen ununterscheidbar sind, folgt direkt daraus, dass wir $\mathcal{S}_{Bad}$ dafür wiederverwenden, also müssen wir nur noch zeigen, dass die Views von $\{P_{j \in BadHub}\}$ damit konsistent sind. Die Views von $P_2$ und $P_3$ werden aus den tatsächlichen Eingaben und Zufallsbänder der korrumpierten Sensoren, die an sie ihre Messungen senden, bzw. aus frischen Nullchiffraten für ehrliche Sender erzeugt. Damit stellen wir sicher, dass die gesamte interne Kommunikation unter den korrumpierten Parteien identisch simuliert wird, wie sie im realen Protokoll abläuft, während die Nachrichten „von außen“ dank der IND-CPA-Eigenschaft der Verschlüsselung vom Realen zumindest unterscheidbar verteilt sind.

Für die View von $P_1$ müssen wir sicherstellen, dass wenn ein oder beide anderen Hubs korrumpiert sind, ihre simulierten gesendeten Nachrichten mit ihren simulierten empfangenen konsistent sind. Dazu wenden wir einfach die gleiche Eval-Funktionalität auf ihre (ggf. teilweise gefälschte) empfangenen Nachrichten an, wie es die ehrlichen Hubs tun würden. Wenn die Zwischenhubs ehrlich sind, simulieren wir die Nachrichten von ihnen stattdessen mit Nullchiffraten, denn selbst wenn alle anderen Sensoren korrumpiert sind, hat weder der Simulator, noch der Unterscheider Zugriff auf die Eingaben und den Zufall der Hubs, und damit ist ein frisches Nullchiffrat sowohl mit den anderen Nachrichten konsistent, als auch von dem realen Eval-Ergebnis ununterscheidbar (s. Überlegungen zu $\mathcal{S}_1$).

Wir haben nun für jede per Sicherheitsdefinition erlaubte Korruptionsstrategie einen Simulator konstruiert, dessen Ausgabe von der jeweiligen View aller korrumpierten Parteien komplexitätstheoretisch ununterscheidbar ist. Dadurch haben wir bewiesen, dass das Protokoll $\pi_{\text{IF}}$ die Funktionalität $f_{\text{IF}}$ in der Anwesenheit der statischen semiehrlichen PPT-Angreifern unter den besagten Annahmen sicher berechnet. □

## 4.4 Komplexitätsanalyse

**Rechenaufwand** Der Rechenaufwand der einzelnen Sensoren (nicht Hubs) im Protokoll 4.1 besteht hauptsächlich aus der Verschlüsselung ihrer Eingaben und beträgt jede Kommunikationsrunde $L$ Enc-Aufrufe für den Informationsvektor und $L^2$ für die Informationsmatrix, wobei $L$ hier die Größe des Systemzustandsvektor ist. Der Gesamtrechenaufwand eines Sensors wird also von $K(L^2+L)$ Enc-Aufrufe dominiert. Der Rechenaufwand zur Berechnung der Informationsvektoren und -matrizen (v. a. die Kovarianzmatrixinversion) kommt zusätzlich dazu.

Die beiden Zwischenhubs haben den gleichen Verschlüsselungsaufwand wie die normalen Sensoren, führen aber zusätzlich die Datenaggregation durch. Diese besteht aus $L^2 + L$ Eval-Aufrufe (um jeweils zwei Skalarchiffrate zu addieren) pro verbundenen Sensor pro Runde. Der zusätzliche Aufwand von $P_2$ beträgt also $|\mathcal{N}_2| \cdot (L^2 + L)$ und der von $P_3$ entsprechend $|\mathcal{N}_3| \cdot (L^2 + L)$ Eval-Aufrufe pro Runde.

Der zentrale Hub hat ebenfalls den gleichen Verschlüsselungsaufwand, aber sein Eval-Aufwand ist kleiner und beträgt lediglich $2(L^2 + L)$ Aufrufe pro Runde.

Der Agent führt einmalig in der Setup-Phase KeyGen aus und ruft dann jede Kommunikationsrunde die Dec-Funktionalität $L^2 + L$ Mal auf, um die aggregierte Informationsvektoren und -matrizen wieder zu entschlüsseln. Dazu kommt der Aufwand zum lokalen Filtern und Prädizieren des Systemzustands.

**Netzwerkaufwand** Der Agent sendet in der Setup-Phase einmalig sein $pk$ an alle $N$ Sensoren und Hubs. Jeder Sensor und Hub sendet pro Kommunikationsrunde $L^2+L$ Chiffrate an den jeweils nächsten Hub bzw. an den Agent, egal ob die Chiffrate frisch oder evaluiert sind. Der Netzwerkaufwand einzelner Parteien pro Runde bleibt in der Größe des Sensornetzwerks konstant. In einer sicheren Implementierung mit dem Paillier-Schema beträgt die Größe eines Chiffrats ca. 500 Byte (s. Seite 19), d. h. mit $L = 4$ (ein einfaches kinematisches 2D-Modell) ist der Netzwerkaufwand also etwa 10 kB pro Sensor pro Runde.

**Mögliche Optimierungen** Weil die Kovarianzmatrizen und ihre Inversen (d. h. Informationsmatrizen) symmetrisch sind, müssen die Sensoren nicht die komplette Matrix verschlüsseln und senden, sondern nur die Werte auf und über ihrer Hauptdiagonale. An dem Protokoll, seiner Korrektheit und seiner Sicherheit ändert diese Veränderung nichts, denn die elementweise Verschlüsselung, Entschlüsselung und homomorphe Addition funktionieren weiterhin gleich. Damit lässt sich aber der Verschlüsselungs- und Evaluierungsaufwand von $L^2+L$ Aufrufe auf $\frac{L(L+1)}{2}+L = \frac{L(L+3)}{2}$ und den Netzwerkaufwand auf 7 kB (mit $L = 4$) pro Runde reduzieren (beide wachsen aber nach wie vor quadratisch in der Größe des Zustandsvektors).

Ein weiterer Optimierungsansatz wäre die „Chiffrat-Verdichtung“ (engl. „ciphertext packing“ [69, 39]), bei der mehrere Werte, z. B. eines Arrays (Vektor bzw. Matrix), in ein Chiffrat gepackt und bei der homomorphen Evaluierung gleichzeitig verarbeitet werden. Diese Möglichkeit bietet uns das Paillier-Schema aber nicht an, und eine naive Implementierung, z. B. durch einfache Konkatenierung der $L$ Array-Elemente der Bitlänge $l$ zu einem $L \cdot l$ Bit langem Klartext, würde spätestens bei der Kodierung und Verschlüsselung negativer Zahlen nach der Def. 2.11 scheitern, denn diese

beansprucht nicht nur $l$ Bit, sondern die komplette Klartextlänge.

## 4.5 Erweiterungen des Grundprotokolls

### 4.5.1 Zusätzliche Hubs

Aus der Komplexitätsanalyse wird deutlich, dass im Protokoll 4.1, wie es aktuell definiert ist, die Hauptrechenlast auf den beiden Zwischenhubs $P_2, P_3$ liegt, die die verschlüsselten Messungen aller anderen Sensoren für $P_1$ voraggregieren müssen. Um diese Last gleichmäßiger zu verteilen, könnten wir das Netzwerk um zusätzliche Hubs erweitern. Dazu gibt es mindestens zwei Ansätze:

- Sollte die flache, dreistufige Architektur des Netzwerks mit wenigen leistungsstarken Hubs und vielen leistungsschwächeren Sensoren beibehalten werden, suchen wir nach einer solchen Anzahl $h$ der Zwischenhubs, dass die Anzahl der Sensoren, die jedem Zwischenhub ihre Messungen senden, etwa gleich der Zahl der Zwischenhubs selbst ist, die ihre aggregierten Daten an den Zentralhub weitergeben. Diese Relation lässt sich mit

  $$N = 1 + h + h^2$$

  ausdrücken, oder umformuliert auch

  $$h = \left\lfloor \frac{\sqrt{4N-3}-1}{2} \right\rfloor \approx \left\lfloor \sqrt{N} \right\rfloor. \tag{4.4}$$

- Wird stattdessen eine gleichmäßige Verteilung der Last auf alle Netzwerkknoten angestrebt, kann eine Binärbaumarchitektur eingeführt werden, wo jeder Hub von nur zwei Unterknoten Messungen bekommt (frisch oder evaluiert), sie mit seiner eigenen fusioniert und weitersendet. Dabei haben alle Hubs die gleiche Rechenlast, die nicht von der Gesamtnetzwerkgröße abhängt.

Wir behaupten, dass das Hinzufügen von zusätzlichen Zwischenhubs bzw. ganzen Zwischenhubebenen keine negative Auswirkung auf die bewiesenen Sicherheitsgarantien des Protokolls hat, solange die Annahme b) aus dem Satz 4.2 gilt. Dies beruht darauf, dass für jede Netzarchitektur nach der Def. 4.2 ein Sicherheitsbeweis ohne Veränderungen an der Satzformulierung analog geführt werden kann.

Im Gegenteil verbessern zusätzliche Hubs sogar die Sicherheitsgarantien, indem sie die Folgen einer gleichzeitigen Agent-Hub-Korruption entschärfen. Weil ein solcher Angriff nur die Nachrichteninhalte an den Angreifer preisgibt, die an den korrumpierten Hub gesendet werden, ist die potenzielle Datenpanne desto kleiner, je weniger

Sender er hat, und desto mehr Hubs muss der Angreifer korrumpieren, um an die gleiche Anzahl von Nachrichten zu kommen.

### 4.5.2 Normalisieren des Informationsvektors

Neben den klassischen Angriffen, die explizit die Korruption einer Partei voraussetzen, gibt es auch weitere Möglichkeiten des unbefugten Datenzugriffs, die erst durch die konkrete, kontextbezogene Aufgabenstellung zustande kommen. Das im Abschnitt 4.1 beschriebene Szenario nimmt implizit an, dass jeder Sensor zu jedem Zeitpunkt eine Messung zum Fusionieren hat. Doch in einer realen Applikation ist es häufig nicht der Fall: z. B. hat eine Radaranlage oft eine maximale Detektionsreichweite, die nicht unbedingt den kompletten Netzwerkbereich abdeckt. Kann ein Sensor $j$ zum Zeitpunkt $k$ keine Messung erheben, ist eine naheliegende Lösung, die an unserem Protokoll und somit an unseren Sicherheitsgarantien nichts ändert, einfach die Eingaben $\hat{\underline{i}}_k^j := \underline{0}, \mathbf{I}_k^j := \mathbf{0}$ zu setzen. Die Nullvektoren und -matrizen fließen unverändert in die homomorphen Additionen und verfälschen nicht die Filterergebnisse. Jedoch öffnet diese Lösung die Tür für eine neue Angriffsart, die nicht direkt mit Kryptographie zu tun hat, sich aber trotzdem kryptographisch lösen lässt.

Bezeichnen wir die Anzahl der Sensoren, die zum Zeitpunkt $k$ eine Messung erheben konnten, mit $M_k \leq N$. Wir bemerken zuerst, dass der Agent den aggregierten Informationsvektor $\hat{\underline{i}}_k$ und die Matrix $\mathbf{I}_k$ im Klartext hat, und dass beide nur Summen der einzelnen Sensoreingaben sind. Die Verteilung dieser Eingaben ist dem Agenten zwar unbekannt, aber unter den gut vertretbaren Annahmen, dass die meisten Sensoren im Netzwerk das gleiche Messmodell $\mathbf{H}_k$ haben und dass ihre Kovarianzen um eine $\mathbf{C}_k^z$ verteilt sind, kann er aus dem zentralen Grenzwertsatz darauf schließen, dass $\mathbf{I}_k$ grundsätzlich zu $M_k \cdot \mathbf{H}_k^{\mathrm{T}} \left(\mathbf{C}_k^z\right)^{-1} \mathbf{H}_k$ tendiert. Durch vorsichtiges Manövrieren (ggf. auch vorgeplant) und Beobachten der Differenzen in seiner empfangenen Nachrichten, kann der Agent bzw. ein semiehrlicher Angreifer, der ihn korrumpiert, $M_k$ analytisch abschätzen und daraus z. B. auf die Positionen und die Detektionsreichweite der einzelnen Sensoren schließen.

**Normalisierung der Eingaben** Die beschriebene Angriffsart wird in der Literatur als „Sensitivitätsangriff“ (engl. *sensitivity attack*) bezeichnet [17] und wird in der formalen Kryptographie üblicherweise nicht betrachtet, weil sie die Eigenschaften der Aufgabenstellung (in diesem Fall, der Eingaben) und nicht des Protokolls ausnutzt [49]. Jedoch können wir ein Protokoll entwerfen, das diesen Angriff zumindest ohne Mehrfachkorruption unmöglich macht. Dazu muss das Sensornetzwerk die aggregierten Informationsvektoren und -matrizen vor der Weitergabe an den Agenten

einfach mit $M_k$ dividieren. Weil aber kein partiell homomorphes Verschlüsselungsschema die Division mit natürlichen Zahlen unterstützt, muss diese Normierung bereits vor dem Verschlüsseln der Eingaben einzelner Sensoren stattfinden, was bedeutet, dass sie sich im Vorfeld sicher verständigen müssen, wie viele Messungen jede Kommunikationsrunde fusioniert werden.

Dazu schlagen wir die ideale Funktionalität

$$f_{\mathrm{Cnt}}\left(E_1, E_2, \ldots, E_N\right) = (M, M, \ldots, M)$$

vor, wobei die Eingaben $E_j$ gleich wie in der Def. 4.1 sind und die gemeinsame Ausgabe $M = (M_1, M_2, \ldots, M_K)$ ist. Das Protokoll 4.2 ($\pi_{\mathrm{Cnt}}$) auf der Seite 56 soll sie sicher implementieren.

***Satz 4.3*** *Unter der Annahme, dass* (KeyGen, Enc, Dec, Eval) *ein asymmetrisches Verschlüsselungsschema mit additivem Homomorphismus und perfekter Korrektheit ist, implementiert das Protokoll $\pi_{\mathrm{Cnt}}$ die Funktionalität $f_{\mathrm{Cnt}}$ korrekt, d. h. für alle $k \in \{1, \ldots, K\}$ ist $M_k$ die Anzahl der Parteien, deren Informationsvektoren in der Runde $k$ nicht Null sind.*

Beweisskizze. Die Korrektheit folgt daraus, dass der Wert $m_k^j$ im Schritt 3 des Protokolls $\pi_{\mathrm{Cnt}}$ auf 1 gesetzt wird, wenn $\hat{\underline{i}}_k^j \neq \underline{0}$, und ansonsten auf 0. Die Summe $\sum_{j=1}^{N} m_k^j$ ist deshalb gleich der Anzahl der Parteien, deren Informationsvektoren in der Runde $k$ nicht Null sind. Und dass für alle $k \in \{1, \ldots, K\}$ die Protokollausgabe

$$M_k = \mathsf{Dec}(sk, [\![m_k]\!]) = \sum_{j=1}^{N} m_k^j$$

ist, kann analog dem Beweis 4.1 hergeleitet werden. □

***Satz 4.4*** *Unter den Annahmen, dass a)* (KeyGen, Enc, Dec, Eval) *ein IND-CPA sicheres asymmetrisches Verschlüsselungsschema mit komplexitätstheoretisch starkem additivem Homomorphismus ist, und dass b) kein Angreifer $P_N$ gleichzeitig mit $P_1, P_2$ oder $P_3$ korrumpieren kann, berechnet $\pi_{\mathrm{Cnt}}$ die Funktionalität $f_{\mathrm{Cnt}}$ in der Anwesenheit der statischen semiehrlichen PPT-Angreifern sicher.*

Beweisskizze. Die Vertraulichkeitsbeweis wird analog dem Beweis des Satzes 4.2 hergeleitet und wird im Folgenden nur skizziert. Der primäre Unterschied zwischen $\pi_{\mathrm{Cnt}}$ und $\pi_{\mathrm{IF}}$ ist, dass im ersteren einer der Sensoren statt dem Agenten als die schlüsselgenerierende Partei fungiert und das Protokollergebnis an alle anderen zur Ausgabe verteilt. Dies führt zu zwei wesentlichen Veränderungen an den Views (außer der offensichtlichen Abwesenheit von $P_{N+1}$):

**Protokoll 4.2 (Sicheres Zählen der Eingaben $\pi_{\mathrm{Cnt}}$)**

- **Parameter**: Alle Parteien kennen (KeyGen, Enc, Dec, Eval), die Netzwerkgröße $N$, die Rundenzahl $K$ und den Sicherheitsparameter $n$.
- **Eingaben**: Jede Partei $P_j$ mit $j \in \{1, \ldots, N\}$ hat eine Sequenz $\hat{\underline{i}}_1^j, \mathbf{I}_1^j, \hat{\underline{i}}_2^j, \mathbf{I}_2^j$, ..., $\hat{\underline{i}}_K^j, \mathbf{I}_K^j$, einen Wert $\mathcal{H}_j = P_{MyHub(j)}$ und eine Indexmenge $\mathcal{N}_j = MySenders(j)$.
- **Das Protokoll**:
  1. $P_N$ berechnet $(pk, sk) \leftarrow \mathsf{KeyGen}\,(1^n)$ und sendet $pk$ an $P_1, \ldots, P_{N-1}$.
  2. Alle Parteien setzen eine interne Variable $k \coloneqq 1$.
  3. Jede $P_{j\in\{1,\ldots,N\}}$ berechnet
  $$[\![m_k^j]\!] \leftarrow \begin{cases} \mathsf{Enc}(pk, 0), & \text{wenn } \hat{\underline{i}}_k^j = \underline{0} \\ \mathsf{Enc}(pk, 1), & \text{sonst} \end{cases}.$$
  4. Jede $P_{j\in\{4,\ldots,N\}}$ sendet $[\![m_k^j]\!]$ an ihren jeweiligen Hub $\mathcal{H}_j$.
  5. Jede $P_{j\in\{2,3\}}$ berechnet für alle $s_1, s_2, \ldots \in \mathcal{N}_j$
  $$[\![m_k^{j+}]\!] \leftarrow [\![m_k^j]\!] +_{pk} [\![m_k^{s_1}]\!] +_{pk} [\![m_k^{s_2}]\!] +_{pk} \ldots$$
  und sendet ihr jeweiliges $[\![m_k^{j+}]\!]$ an $\mathcal{H}_j = P_1$.
  6. $P_1$ berechnet $[\![m_k]\!] \leftarrow [\![m_k^1]\!] +_{pk} [\![m_k^{2+}]\!] +_{pk} [\![m_k^{3+}]\!]$ und sendet $[\![m_k]\!]$ an $P_N$.
  7. $P_N$ berechnet $M_k \coloneqq \mathsf{Dec}(sk, [\![m_k]\!])$ und sendet $M_k$ im Klartext an $P_1, \ldots, P_{N-1}$.
  8. Alle Parteien geben $M_k$ aus, setzen $k \coloneqq k+1$ und, wenn $k \leq K$, springen zum Schritt 3 zurück.

- $\text{view}_N^{\pi_{\text{Cnt}}}(E_1, E_2, \ldots, E_N)$ enthält in jeder Kommunikationsrunde zusätzlich das Chiffrat $[\![m_k]\!]$ statt $pk$ (den sie selbst erzeugt). Dieses ist aber leicht zu simulieren, indem $\mathsf{Enc}(pk, M_k)$ einfach mit unabhängigem Zufall ausgeführt wird, denn $M_k$ ist Teil der Ausgabe von $P_N$ und somit der Eingabe vom Simulator.
- Alle anderen Views enthalten jede Runde zusätzlich den Wert $M_k$ im Klartext, dieser ist aber trivial simulierbar, weil er auch in ihren Ausgaben enthalten ist: der Simulator gibt ihn einfach unverändert wieder aus.

Die formale Sicherheit des Protokolls beruht letztendlich darauf, dass die homomorphe Aufsummierung der Nachrichten von anderen Parteien ausgeführt wird, als die Entschlüsselung und die Verteilung des Ergebnisses im Schritt 8. □

$P_N$ wurde hier als die schlüsselerzeugende Partei nur wegen der einfacheren Schreibweise gewählt, und nicht weil sie gegenüber $P_4, \ldots, P_{N-1}$ einen Sicherheitsvorteil bringen würde. Der Sicherheitsbeweis würde analog für jede andere Nicht-Hub-Partei funktionieren. Das Endergebnis der Berechnung wird an alle Parteien im Klartext übertragen, weil es formal kein Geheimnis ist, wenn alle Parteien es kennen.

Bisher waren die Protokolle 4.1 ($\pi_{\text{IF}}$) und 4.2 ($\pi_{\text{Cnt}}$) unabhängig voneinander definiert. Als letzter Schritt müssen wir sie kombinieren und ihre Sicherheit unter der sequentiellen Komposition zu beweisen. Dazu definieren wir eine neue ideale Funktionalität

$$f_{\text{nIF}}(E_1, E_2, \ldots, E_N, \lambda) = \left(M, M, \ldots, M, A'_{N+1}\right),$$

die die Funktionalitäten $f_{\text{IF}}$ und $f_{\text{Cnt}}$ in sich vereint, wobei die Ausgabe der Sensoren nun $M$ aus $f_{\text{Cnt}}$ enthält und die Ausgabe $A'_{N+1}$ des Agenten hier im Gegensatz zu $A_{N+1}$ in $f_{\text{IF}}$ aus *normierten* Informationsvektoren bzw. Nullvektoren besteht, sodass für alle $k \in \{1, \ldots, K\}$:

$$a'_k \coloneqq \begin{cases} (\underline{0}, \mathbf{0}), & \text{wenn } M_k = 0 \\ \left(\frac{1}{M_j}\hat{\underline{i}}_j, \frac{1}{M_j}\mathbf{I}_j\right), & \text{sonst} \end{cases},$$

$$A'_{N+1} \coloneqq (a'_1, a'_2, \ldots, a'_K). \tag{4.5}$$

Im ersten Schritt stellen wir das Protokoll 4.3 ($\pi'_{\text{nIF}}$) auf der Seite 58 auf, das die Berechnung von $f_{\text{nIF}}$ im Hybridmodell nach der Def. 2.9 vertraulich auf $f_{\text{Cnt}}$ und $f_{\text{IF}}$ reduziert. Anschließend ersetzen wir die Orakelaufrufe darin durch Ausführungen der Protokolle $\pi_{\text{Cnt}}, \pi_{\text{IF}}$ und benutzen den sequentiellen Kompositionssatz, um die passive Sicherheit des resultierenden Protokolls $\pi_{\text{nIF}}$ herzuleiten.

***Satz 4.5*** *Das Protokoll $\pi'_{\text{nIF}}$ implementiert die Funktionalität $f_{\text{nIF}}$ korrekt.*

Beweisskizze. Die Korrektheit der Ausgaben von $P_j$ mit $j \in \{1, \ldots, N\}$ folgt direkt aus dem Satz 4.3. Die Korrektheit der Ausgabe von $P_{N+1}$ folgt aus dem

**Protokoll 4.3 (Sicheres verteiltes Informationsfilter mit Normalisierung $\pi'_{\mathrm{nIF}}$ im Hybridmodell)**

- **Parameter**: Alle Parteien kennen die Netzwerkgröße $N$, die Rundenzahl $K$ und den Sicherheitsparameter $n$.
- **Eingaben**:
  - Jede Partei $P_j$ mit $j \in \{1, \ldots, N\}$ (Sensor) hat eine Sequenz $\hat{\underline{i}}_1^j, \mathbf{I}_1^j, \hat{\underline{i}}_2^j, \mathbf{I}_2^j$, ..., $\hat{\underline{i}}_K^j, \mathbf{I}_K^j$, einen Wert $\mathcal{H}_j = P_{MyHub(j)}$ und eine Indexmenge $\mathcal{N}_j = MySenders(j)$.
  - Die Partei $P_{N+1}$ (Agent) hat keine bzw. eine leere Eingabe $\lambda$.
- **Orakel**: Alle Parteien haben Zugriff auf die Orakelfunktionalitäten $f_{\mathsf{Cnt}}$, wie oben definiert, und $f_{\mathsf{IF}}$ nach der Def. 4.1.
- **Das Protokoll**:
  1. Jede $P_{j\in\{1,\ldots,N\}}$ sendet ihre Eingabe $E_j$ an das Orakel $f_{\mathsf{Cnt}}$.
  2. Die Parteien $\{P_j : j \in \{1, \ldots, N\}\}$ rufen das Orakel $f_{\mathsf{Cnt}}$ auf, um die gemeinsame Ausgabe $M = (M_1, M_2, \ldots, M_K)$ zu erhalten.
  3. Jede $P_{j\in\{1,\ldots,N\}}$ berechnet lokal für $k \in \{1, \ldots, K\}$

$$\underline{x}_k^j := \begin{cases} \underline{0}, & \text{wenn } M_k = 0 \\ \frac{1}{M_k}\hat{\underline{i}}_k^j, & \text{sonst} \end{cases} \quad \text{sowie}$$

$$\mathbf{X}_k^j := \begin{cases} \mathbf{0}, & \text{wenn } M_k = 0 \\ \frac{1}{M_k}\mathbf{I}_k^j, & \text{sonst} \end{cases}$$

     und setzt $E'_j := \left(\underline{x}_1^j, \mathbf{X}_1^j, \underline{x}_2^j, \mathbf{X}_2^j, \ldots, \underline{x}_K^j, \mathbf{X}_K^j\right)$.
  4. Jede $P_{j\in\{1,\ldots,N\}}$ sendet $E'_j$ an das Orakel $f_{\mathsf{IF}}$.
  5. Alle Parteien rufen das Orakel $f_{\mathsf{IF}}$ auf, das $A'_{N+1}$ an $P_{N+1}$ zurücksendet.
  6. Jede $P_j$ mit $1 \leq j \leq N$ gibt $M$ aus. $P_{N+1}$ gibt $A'_{N+1}$ aus.

Satz 4.1 und der Distributivität der Skalarmultiplikation über Matrizenaddition. Bezeichnen wir das erste Element der Ausgabe des Orakels $f_{\mathrm{IF}}$ an $P_{N+1}$ im $\pi_{\mathrm{nIF}}$ mit $\hat{\underline{i}}'_1$, dann gilt:

$$\hat{\underline{i}}'_1 = \sum_{j=1}^{N} \underline{x}_1^j = \sum_{j=1}^{N} \frac{1}{M_k} \hat{\underline{i}}_k^j = \frac{1}{M_k} \sum_{j=1}^{N} \hat{\underline{i}}_k^j = \frac{1}{M_k} \hat{\underline{i}}_1.$$

Die gleiche Überlegung gilt ebenfalls für alle anderen Elemente der Orakelausgabe, was uns letztendlich die Gleichung (4.5) herleiten und damit die Korrektheit des Protokolls zeigen lässt. □

***Satz 4.6*** *Das orakelunterstützte Protokoll $\pi'_{\mathrm{nIF}}$ reduziert die Funktionalität $f_{\mathrm{nIF}}$ vertraulich auf die Funktionalitäten $f_{\mathrm{Cnt}}$ und $f_{\mathrm{IF}}$ im Hybridmodell.*

Beweis. Zuerst betrachten wir den Fall, dass die Partei $P_1$ korrumpiert ist. Die Partei hat die Eingabe $E_1$ und Ausgabe $M$. Sie empfängt eine Nachricht vom Orakel $f_{\mathrm{Cnt}}$, nämlich $M$. Der Simulator $\mathcal{S}_1$ bekommt als Eingabe $(1^n, E_1, M)$ und gibt einfach $(E_1; M)$ aus. Weil das Protokoll im Hybridmodell deterministisch abläuft, muss $\mathcal{S}_1$ kein Zufallsband erzeugen. Aus dem gleichen Grund ist seine Ausgabe immer genauso wie die Sicht der Partei aufs reale Protokoll „verteilt“ (und ist somit trivialerweise davon ununterscheidbar). Dadurch haben wir gezeigt, dass

$$\{\mathcal{S}_1\,(1^n, E_1, M)\} \stackrel{\mathrm{c}}{\equiv} \{\mathrm{view}_1^{\pi'_{\mathrm{nIF}}}\,(E_1, E_2, \ldots, E_N)\}.$$

Für die Parteien $P_2, \ldots, P_N$ erfolgt die Simulatorkonstruktion analog, weil sie sich im Hybridmodell genauso verhalten, wie $P_1$.

Nun betrachten wir den Fall, dass die Partei $P_{N+1}$ korrumpiert wird. Der Agent hat eine leere Eingabe und die Ausgabe $A'_{N+1}$. Er empfängt eine Nachricht vom Orakel $f_{\mathrm{IF}}$, nämlich $A'_{N+1}$. Der Simulator $\mathcal{S}_{N+1}$ bekommt als Eingabe $\left(1^n, A'_{N+1}\right)$ und gibt einfach $\left(A'_{N+1}\right)$ aus. Analog $\mathcal{S}_1$ können wir darauf schließen, dass

$$\{\mathcal{S}_{N+1}\left(1^n, A'_{N+1}\right)\} \stackrel{\mathrm{c}}{\equiv} \{\mathrm{view}_{N+1}^{\pi'_{\mathrm{nIF}}}\,(E_1, E_2, \ldots, E_N)\}.$$

Den Fall der Mehrfachkorruption müssen wir nicht gesondert beachten, weil die Parteien im Hybridmodell zu keinem Zeitpunkt miteinander kommunizieren, sondern nur mit den Orakeln. Damit haben wir gezeigt, dass das Protokoll $\pi'_{\mathrm{nIF}}$ die Funktion $f_{\mathrm{nIF}}$ im Hybridmodell privat auf die Funktionen $f_{\mathrm{Cnt}}$ und $f_{\mathrm{IF}}$ reduziert. □

Als letztes definieren wir ein Protokoll $\pi_{\mathrm{nIF}}$, das genauso wie $\pi'_{\mathrm{nIF}}$ abläuft, aber die Orakel-Aufrufe von $f_{\mathrm{Cnt}}$ und $f_{\mathrm{IF}}$ durch die Ausführung von $\pi_{\mathrm{Cnt}}$ (Protokoll 4.1) bzw. $\pi_{\mathrm{IF}}$ (Protokoll 4.2) ersetzt.

***Satz 4.7*** *Unter den Annahmen, dass a)* (KeyGen, Enc, Dec, Eval) *ein IND-CPA sicheres asymmetrisches Verschlüsselungsschema mit komplexitätstheoretisch starkem*

*additivem Homomorphismus ist, und b) kein Angreifer $P_N$ oder $P_{N-1}$ gleichzeitig mit $P_1, P_2$ oder $P_3$ korrumpieren kann, berechnet $\pi_{\text{nIF}}$ die Funktionalität $f_{\text{nIF}}$ in der Anwesenheit der statischen semiehrlichen PPT-Angreifern sicher.*

BEWEISSKIZZE. Die Vertraulichkeit des Protokolls $\pi_{\text{nIF}}$ folgt direkt aus seiner Definition, dem Satz 4.6 und dem sequentiellen Kompositionssatz [35, Satz 7.5.7]. Weil es im $\pi_{\text{nIF}}$ zwei schlüsselerzeugende Parteien gibt, werden beide explizit in der Annahme b) referenziert. □

Unter der sequentiellen Komposition wie in $\pi'_{\text{nIF}}$ und $\pi_{\text{nIF}}$ ist es erforderlich, dass die Messungen für alle Zeitrunden bereits vor der Protokollausführung vorliegen, weil alle $M_1, \ldots, M_K$ berechnet werden müssen, bevor die ersten eigentlichen Messungen ausgetauscht werden. Dies ist in den realen Applikationen natürlich nicht gegeben, wenn die Messungen in Echtzeit erhoben und ausgezählt werden.

Um die Realität (bzw. unsere eigene Simulation im Abschnitt 4.6) genauer zu formalisieren, müssten wir das Protokoll $\pi_{\text{nIF}}$ durchs „Spleißen“ der beiden Protokolle 4.1 und 4.2 herleiten, indem wir die Setup-Phase von $\pi_{\text{IF}}$ um die Schlüsselerzeugung bei $P_N$ erweitern und die Schritte 3 bis 7 des $\pi_{\text{Cnt}}$ vor dem Schritt 4 des $\pi_{\text{IF}}$ einfügen. Weil der Sicherheitsbeweis für diese Variante aber analog den Beweisen 4.2 und 4.4 erfolgen würde, führen wir ihn hier nicht weiter aus.

**Renormalisierung** Eine Normalisierung wie oben beschrieben verfälscht die aggregierten Informationsformen der Messungen insofern, dass die letzteren beim Filtern (2.6a) und (2.6b) um Faktor $M_k$ weniger in das Filterergebnis eingehen, was sich auf die Schätzungsgenauigkeit negativ auswirken kann. Intuitiv, wenn der Agent viele Messungen seiner Position hat, ist es vorteilhaft, das Aggregat davon beim Filtern entsprechend hoch zu gewichten. Weil wir aber dem Agenten keine genaue Anzahl der Messungen preisgeben wollen, können wir einfach die normalisierten Informationsvektoren und -matrizen mit einem zeitinvarianten Gewichtungsfaktor multiplizieren. Der Erwartungswert $\mathrm{E}\{M_k\}$ (integriert über das gesamte Feld) würde als Gewichtungsfaktor die Verfälschung beim Filtern minimieren und dabei keine Informationen über die aktuelle $M_k$ an den Agenten verraten.

Diese Renormalisierung kann bereits im Schritt 3 des Protokolls 4.3 stattfinden, indem wir den Term $\frac{1}{M_k}$ durch $\frac{\mathrm{E}\{M_k\}}{M_k}$ ersetzen, oder erst am Schluss, indem der Zentralhub die aggregierten Chiffrate homomorph mit dem Gewichtungsfaktor multipliziert, bevor er sie an den Agenten sendet. Der erste Ansatz hätte zufolge, dass der $\mathrm{E}\{M_k\}$ allen Sensoren explizit als Eingabe gegeben werden müsste, und würde einen Genauigkeitsvorteil bringen, weil die normalisierten Werte grundsätzlich größer wären und

durch die Quantisierung weniger Information verlieren würden. Für den zweiten Ansatz müsste nur der Zentralhub den $\mathrm{E}\{M_k\}$ kennen und der Erwartungswert müsste auf den nächsten Integer gerundet werden (außerdem müssten wir die Funktionalität $f_{IF}$ neu formulieren, das Protokoll $\pi_{\mathrm{IF}}$ um einen zusätzlichen Schritt erweitern, und das komplette Sicherheitsbeweis erneut führen).

## 4.6 Genauigkeitsauswertung mittels numerischer Simulation

Aus der perfekten Korrektheit des Verschlüsselungschemas folgt, dass die Verschlüsselung und die Entschlüsselung der Daten keine Auswirkung auf die Genauigkeit unserer Verfahren haben können. Ein Genauigkeitsverlust gegenüber einer unverschlüsselten Ausführung kann also nur durch die Quantisierung der Messdaten entstehen. Die strikte theoretische Grenze davon wurde im Abschnitt 2.3 angegeben, jedoch ist es viel schwieriger, eine derart präzise Aussage über die Genauigkeit eines gesamten Filterverfahren zu machen, das die quantisierten Daten benutzt. Um die Genauigkeit der Ausgaben der Protokolle $\pi_{\mathrm{IF}}$ und $\pi_{\mathrm{nIF}}$ zu evaluieren, haben wir deshalb eine Softwaresimulation in der Programmiersprache *Python* mit dem kostenlosen Programmpaket *NumPy/SciPy* entwickelt. Ihr Quellcode ist öffentlich unter `https://github.com/KIT-ISAS/PPIF` zugänglich.

**Simuliertes Szenario** Wie in Abb. 4.2 dargestellt, bewegt sich ein mobiler Agent in einer 2D Ebene über ein $100\,\mathrm{m} \times 100\,\mathrm{m}$ Feld, auf dem 25 Radarstationen (Sensoren) verteilt sind. Die Sensoren sind in einem $5 \times 5$ Raster angeordnet und hierarchisch miteinander verbunden. Gemäß der Formel (4.4) fungieren 4 Sensoren als Zwischenhubs, um die Messungen von jeweils 5 anderen vozuraggregieren und an den zentralen Hub zu senden, der mit dem Agenten kommuniziert.

Der simulierte Agent wird am Anfang eines Simulationslaufs an einen Randpunkt des Felds, der zufällig und gleichverteilt gezogen wird, mit einem Geschwindigkeitsvektor, der aus $\mathcal{N}(\underline{0}, \mathbf{C}^w)$ gezogen wird, gesetzt. Dieser Vektor bleibt konstant, bis der Agent das Feld verlässt und damit den aktuellen Simulationslauf beendet. Der Agent modelliert in seinem Systemzustand $\underline{\boldsymbol{x}}_k$ nur seine Position zum Zeitpunkt $k$ (Systemzustandsgröße $L = 2$) und führt jede Zeitrunde eine Zustandsprädiktion nach den Formeln (2.1a) und (2.1b) durch, wobei

$$\mathbf{A}_k = \mathbf{A} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad \text{und} \quad \mathbf{C}_k^w = \mathbf{C}^w = \begin{bmatrix} \sigma_v^2 & 0 \\ 0 & \sigma_v^2 \end{bmatrix}$$

(mit einer einstellbaren, aber zeitinvarianten $\sigma_v$) sind. Damit reduziert sich der lokale

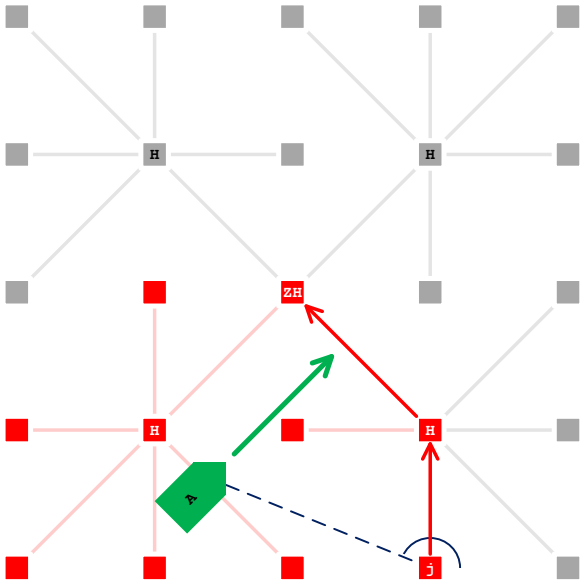


**Abbildung 4.2:** Räumlicher Aufbau der Simulation des Protokolls 4.1. Die roten Knoten stellen die Sensoren dar, die den Agenten in dem aktuellen Zeitschritt erfassen können; die grauen sind dafür zu weit entfernt und senden nur einen Nullvektor.

„Prädiktionsschritt“ auf:

$$\begin{aligned}\hat{\underline{x}}^{\mathrm{p}}_{k+1} &:= \hat{\underline{x}}^{\mathrm{e}}_{k} \\ \mathbf{C}^{\mathrm{p}}_{k+1} &:= \mathbf{C}^{\mathrm{e}}_{k} + \mathbf{C}^{w}\end{aligned}$$

Die simulierten Sensoren „erfassen“ jede Zeitrunde die Position des Agenten, wenn er innerhalb einer (einstellbaren) Distanz $r_{\max}$ von dem jeweiligen Sensor ist. Die relative Position des Agenten bzgl. des Sensors wird in die Polarkoordinaten (Radius $r$ und Azimut $\theta$) umgerechnet und mit additiven Rauschen aus $\mathcal{N}\left(\underline{0}, \left[\begin{smallmatrix}\sigma_r^2 & 0\\ 0 & \sigma_\theta^2\end{smallmatrix}\right]\right)$ (mit einstellbaren $\sigma_r, \sigma_\theta$) verrauscht, um das Messrauschen einer Radarstation zu simulieren. Die verrauschte Messung wird dann wieder in die globale kartesische Koordinaten als $\hat{\underline{z}}^{j}_{k}$ umgerechnet. Die Messkovarianz in kartesischen Koordinaten wird über die Jakobi-Matrix der Polartransformation berechnet:

$$\mathbf{J}_k := \begin{bmatrix}\cos(\theta_k) & -r_k \sin(\theta_k)\\ \sin(\theta_k) & r_k \cos(\theta_k)\end{bmatrix} \quad \text{und} \quad \mathbf{C}^{z}_{k} := \mathbf{J}_k \begin{bmatrix}\sigma_r^2 & 0\\ 0 & \sigma_\theta^2\end{bmatrix} \mathbf{J}_k^{\mathrm{T}}.$$

Anschließend werden die Messung $\hat{\underline{z}}^{j}_{k}$ und ihre Kovarianzmatrix vom Sensor $j$ in ihre Informationsformen $\hat{\underline{i}}^{j}_{k}, \mathbf{I}^{j}_{k}$ nach den Formeln (2.5a) bzw. (2.5b) umgewandelt, mit einer einstellbaren Präzisionsstufe $f$ nach der Def. 2.10 in Fixkommazahlen quantisiert, und mit dem öffentlichen Schlüssel $pk$ des Agenten verschlüsselt[1], um

1 Zwecks kürzerer Laufzeit der Simulationssoftware wird der Schlüssel nicht für jeden Agenten neu erzeugt, sondern hart im Code verdrahtet, was selbstverständlich für eine reale Applikation undenkbar wäre. Seine Länge (64 Bit) entspricht ebenfalls in keinem Fall den Sicherheitsempfehlungen.

letztendlich im Protokoll $\pi_{\mathrm{IF}}$ verwendet zu werden. Wenn der Sensor keine Messung erheben konnte, weil der Agent zu weit entfernt war, sendet er stattdessen zwei Nullchiffrat-Arrays der entsprechenden Größen.

Um gleichzeitig die Genauigkeit des Protokolls $\pi_{\mathrm{nIF}}$ auszuwerten, wird parallel eine mit dem Term $\frac{\mathrm{E}\{M_k\}}{M_k}$ (re)normalisierte Eingabe mitversendet. Zur Ermittlung der $M_k$ wird eine Ausführung von $\pi_{\mathrm{Cnt}}$ simuliert. $\mathrm{E}\{M_k\}$ kann offline durch das Integrieren von $M_k\,(\underline{x})$ über alle mögliche Systemzustände $\underline{x}$ berechnet werden. Weil die genaue Vorgehensweise dabei vom konkreten Aufbau des Sensornetzwerk und $r_{\max}$ stark abhängig ist, wird sie hier aus Platzgründen nicht ausführlich beschrieben.

Der Agent bekommt in beiden Protokollen jede Runde einen aggregierten Informationsvektor $\hat{\underline{i}}_k$ und Informationsmatrix $\mathbf{I}_k$ und fusioniert sie nach den Formeln (2.6a) und (2.6b) lokal mit seinem prädizierten Zustand und Kovarianz, die er davor in ihre Informationsformen $\hat{\underline{y}}_k^{\mathrm{p}}, \mathbf{Y}_k^{\mathrm{p}}$ nach den Formeln (2.4a) und (2.4b) umwandelt. Anschließend werden die gefilterten Informationsformen $\hat{\underline{y}}_k^{\mathrm{e}}, \mathbf{Y}_k^{\mathrm{e}}$ zurück in $\hat{\underline{x}}_k^{\mathrm{e}}, \mathbf{C}_k^{\mathrm{e}}$ konvertiert und nächste Runde für die Prädiktion wiederverwendet.

**Simulationsergebnisse** Um die Genauigkeit der Protokolle $\pi_{\mathrm{IF}}$ und $\pi_{\mathrm{nIF}}$ unter verschiedenen Fixkommapräzisionsstufen $f$ (in Bits) zu testen, wurde die Simulation jeweils 30 000 Mal in drei unterschiedlichen Szenarien ausgeführt:

1. Im „Standardszenario“ galt $\sigma_\theta = 5°, \sigma_r = 2\,\mathrm{m}, r_{\max} = 50\,\mathrm{m}$;
2. Im „günstigen“ Szenario galt $\sigma_\theta = 5°, \sigma_r = 2\,\mathrm{m}, r_{\max} = 200\,\mathrm{m}$;
3. Im „ungünstigen“ Szenario galt $\sigma_\theta = 15°, \sigma_r = 5\,\mathrm{m}, r_{\max} = 50\,\mathrm{m}$.

In allen drei Szenarien war $\sigma_v = 5\,\mathrm{m/s}$. Die evaluierten Präzisionsstufen waren 8 Bit, 16 Bit und 24 Bit. Alle Informationsvektoren und -matrizen wurden parallel als float-Arrays und als Integer-Arrays in jeweils drei Präzisionsstufen an den Agent übergeben, der sie unabhängig voneinander in vier parallelen Schätzerinstanzen für jeweils $\pi_{\mathrm{IF}}$ und $\pi_{\mathrm{nIF}}$ (insgesamt acht) fusionierte. Dadurch wurde sichergestellt, dass die Evaluation der unterschiedlichen Präzisionsstufen in jedem Szenario und in beiden Protokollen mit den genau gleich verrauschten Messdaten erfolgte.

Für das $\pi_{\mathrm{nIF}}$ benötigten wir in den Szenarien 1 und 3 außerdem den Wert von $\mathrm{E}\{M_k\}$. Dieser ist für den gegebenen Aufbau und $r_{\max} = 50\,\mathrm{m}$ gleich der Summe der kreis(segment)förmigen Flächen, die von den 25 Radaren jeweils effektiv erfasst werden, geteilt durch die gesamte Fläche des beobachten Gebiets:

$$\mathrm{E}\{M_k\} = \frac{(11\pi + 3\sqrt{3} + 1) r_{\max}^2}{4 r_{\max}^2} \approx 10.$$

| Präz.-Stufe | Szenario 1: RMSE | | Szenario 2: RMSE | | Szenario 3: RMSE | |
|---|---|---|---|---|---|---|
| | $\pi_{\text{IF}}$ | $\pi_{\text{nIF}}$ | $\pi_{\text{IF}}$ | $\pi_{\text{nIF}}$ | $\pi_{\text{IF}}$ | $\pi_{\text{nIF}}$ |
| float | 1.184 496 | 1.186 199 | 0.830 201 | 0.827 849 | 2.940 632 | 2.955 417 |
| 8 Bit | 1.225 865 | 1.224 356 | 0.854 503 | 0.983 073 | 3.481 097 | 3.474 726 |
| 16 Bit | 1.184 497 | 1.186 200 | 0.830 200 | 0.827 850 | 2.940 658 | 2.955 430 |
| 24 Bit | 1.184 496 | 1.186 199 | 0.830 201 | 0.827 849 | 2.940 632 | 2.955 417 |

**Tabelle 4.1:** Ergebnisse der Evaluation der Protokolle $\pi_{\text{IF}}$ und $\pi_{\text{nIF}}$ in den drei Testszenarien.

Um den Effekt der Renormalisierung auf das Szenario 2 zu testen, nutzten wir auch darin zur Gewichtung den gleichen Erwartungswert. Dabei bemerken wir, dass mit $r_{\max} = 200\,\text{m}$ der tatsächliche Mittelwert $\hat{M}_k = N = 25$ für alle $k$ sein muss und in diesem Fall die Protokolle $\pi_{\text{IF}}$ und $\pi_{\text{nIF}}$ sich funktional identisch verhalten.

Anschließend wurden für jede Präzisionsstufe, Szenario und Protokoll die Wurzel des mittleren quadratischen Fehlers (RMSE) der jeweiligen Schätzungen bzgl. der tatsächlich simulierten Positionen berechnet. Die RMSE-Werte sind in der Tabelle 4.1 dargestellt. Weil die Simulationen nichtdeterministisch waren, wurden in jedem Szenario unterschiedlich viele Schätzungen erstellt, sowie unterschiedliche tatsächliche $\hat{M}_k$ berechnet:

1. Standardszenario: 574 604 Schätzungen, $\hat{M}_k = 9.951$;
2. „Günstiges“ Szenario: 580 381 Schätzungen, $\hat{M}_k = 25$;
3. „Ungünstiges“ Szenario: 569 255 Schätzungen, $\hat{M}_k = 9.962$.

Aus der Evaluation lässt sich erkennen, dass für ausreichend große $f$ der Filterungsfehler mit Quantisierung zu dem Fehler ohne Quantisierung konvergiert, was auch zu erwarten wäre, denn die Mantisse des als Vergleichsgrundlage verwendeten float-Formats selbst nur aus 23 Bit besteht [42]. Auch erwartungsgemäß ist die Genauigkeit des $\pi_{\text{nIF}}$ in den meisten Fällen etwas schlechter als die von $\pi_{\text{IF}}$, weil die Renormalisierung u. U. die aggregierten Informationsvektoren im Filterschritt disproportional stark oder schwach gewichtet, verglichen mit der tatsächlichen Anzahl der darin fusionierten Messungen $M_k$.

Bemerkenswert ist, dass in den Szenarien 1 und 3 mit $f = 8$ das normalisierte Protokoll $\pi_{\text{nIF}}$ auch etwas kleineren Fehler als $\pi_{\text{IF}}$ produziert und mit $f \geq 16$ einen etwas größeren, während im „günstigen“ Szenario 2 es genau umgekehrt ist. Das letztere hängt vermutlich damit zusammen, dass die Normalisierung des aggregierten Informationsvektors seine Gewichtung ggü. dem prädizierten Systemzustand beim Filtern künstlich um Faktor 2.5 verkleinert, was zu ungenaueren Schätzungen führt.

Insgesamt lässt sich eine klare Aussage treffen, dass bereits ab einer 16-Bit-Präzision

die Schätzungen in beiden Protokollen kaum einen Nachteil ggü. der unverschlüsselten Berechnung mit Gleitkommazahlen aufweisen, und dass bei $\mathrm{E}\{M_k\} < N$ $\pi_{\mathrm{nIF}}$ eine praktikable oder sogar genauere Alternative für $\pi_{\mathrm{IF}}$ sein kann.

## 4.7 Bewertung

Im Abschnitt 4.1 haben wir ein ideale Funktionalität für ein verteiltes Informationsfilter aufgestellt und im Abschnitt 4.3 unter bestimmten Annahmen bewiesen, dass das Protokoll 4.1 ($\pi_{\mathrm{IF}}$) diese Funktionalität sicher in der Anwesenheit der statischen semiehrlichen PPT-Angreifer berechnet. Im Abschnitt 4.5.2 haben wir einen analogen Beweis für das um die Normalisierung der Eingaben erweiterte Protokoll 4.3 ($\pi_{\mathrm{nIF}}$) skizziert. Um diese Aussagen in den Kontext zu setzen, müssen wir zuerst genauer auf die getroffenen Annahmen eingehen.

**Sicherheitsannahmen** Für unsere Beweise haben wir das semiehrliche (passive) Angreifermodell gewählt, d. h. angenommen, dass die korrumpierten Parteien sich weiterhin ans Protokoll halten und v. a. ihre tatsächlichen Messungen unverändert verschlüsseln und versenden. Ein Beweis in diesem Modell gibt uns die Sicherheitsgarantie gegen „unbeabsichtigte Datenpannen“ (engl. *inadvertent leakage of information*) [52, Abschn. 6.4.1], d. h. dass kein Horcher, der entweder die Kommunikation zwischen den Parteien abhört oder den RAM einer oder mehreren davon auslesen kann, zusätzliche Informationen über die anderen daraus gewinnen kann.

Die letztere Aussage müssen wir nun insofern relativieren, dass die formale Definitionen der „Datenpannen“ oft kontraintuitiv sind. Im Protokoll 4.2 wird z. B. der Wert $M_k$ im Klartext verteilt, weil alle beteiligten Parteien ihn am Ende lernen sollen und es somit formal kein Geheimnis ist. In einer realen Applikation würde er für die Übertragung natürlich verschlüsselt sein, um ihn vor Horchern an der Leitung zu schützen. Andererseits bemerken wir, dass wenn der Angreifer den Agenten und gleichzeitig einen Sensor (z. B. $P_4$) korrumpiert, er sowohl die Bewegungsdaten vom Agenten, als auch die zugehörigen Werte $M_k$ lernen und damit wieder den Sensitivitätsangriff durchführen kann, der die Hauptmotivation für den Entwurf des Protokolls $\pi_{\mathrm{nIF}}$ war.

Auch die Aussage, dass ein Horcher an der Leitung nichts über die Eingaben der Parteien lernen kann, hat einen Vorbehalt. Aus der Beobachtung, welche Sensoren an welche Hubs ihre Daten senden, kann der Horcher schon nach einer Kommunikationsrunde die Informationen über die Netzwerkarchitektur gewinnen, die in den Eingaben $\mathcal{H}_j, \mathcal{N}_j$ enthalten sind. Dieser Angriff lässt sich verhindern, wenn jeder Sen-

sor nicht nur seine Messung an seinen Hub, sondern auch gleichlange Nullchiffrate an alle anderen Hubs sendet, die sie dann verwerfen. Es ist im Einzelfall abzuwägen, ob der Schutz der Informationen, die oft auch aus der räumlichen Anordnung der Sensoren erlernbar sind, die damit verbundene Steigerung des Netzwerkvolumens und des Rechenaufwands rechtfertigt.

Die Annahme, dass das gewählte Verschlüsselungsschema IND-CPA sicher ist und komplexitätstheoretisch starken Homomorphismus hat, kann nur insofern problematisch werden, dass die IND-CPA-Sicherheit der meisten realen Verschlüsselungsverfahren auf weiteren, unbewiesenen Annahmen basiert. Z. B. ist die Sicherheit des von uns bevorzugten Paillier-Verschlüsselungsschemas nur unter einer ebensolchen Annahme bewiesen worden [64]. Weil die meisten Kryptographieforscher solche Grundannahmen aber für glaubwürdig halten, werden sie selten umstritten. Wir bemerken außerdem, dass die komplexitätstheoretischen Annahmen nur unter der gleichzeitigen Annahme des effizienten (PPT-beschränkten) Angreifer Sinn haben.

Die letzte Annahme, dass der Agent und die Hubs nicht gleichzeitig korrumpierbar sind, ist am schwierigsten zu rechtfertigen, weil wir dadurch den Angreifer und damit unsere Sicherheitsgarantien weiter abschwächen. Der Hintergrund, wieso wir sie benötigen, wurde ausführlich im Abschnitt 4.3.2 erklärt. Eine naheliegende Rechtfertigung für die Annahme ist, dass die wenigen zentralen Knoten des Sensornetzwerks (Hubs) grundsätzlich besser gegen Korruption geschützt sein müssen, als die Mehrheit ihnen untergeordneter Sensoren. Andererseits bemerken wir auch, dass je mehr Hubs es im Netztwerk gibt, desto geringer ist das Ausmaß einer Datenpanne, wenn einer davon korrumpiert wird.

**Aktive Angriffe** Das semiehrliche Angreifermodell ist eine sehr schwache Sicherheitsannahme, die in der Regel nur als Vorstufe zur Sicherheit ggü. stärkeren Angreifern benutzt wird [52]. Bereits wenn ein Angreifer die korrumpierte Parteien minimal aktiv steuern kann (z. B. ihre Eingaben oder Zufallsbänder manipulieren), kann er u. U. das gesamte passiv sichere Protokoll brechen. Grundsätzlich kann jede Funktionalität, die gegen semiehrliche Angreifer sicher implementierbar ist, auch gegen boshafte (aktive) Angreifer sicher gemacht werden [35, Abschn. 7.4], was ein Grund dafür ist, dass die meiste Literatur zur sicheren Signalverarbeitung sich auf das semiehrliche Modell beschränkt [49]. Diese Methode ist aber nicht unmittelbar auf homomorphe Verschlüsselungen anwendbar und wird grundsätzlich im Zusammenhang mit dem *Garbled Circuits* Protokoll von Andrew Yao [77] verwendet [35, 41].

Eine naheliegende Angriffsmöglichkeit, die jeder aktive bzw. verstärkt semiehrliche Angreifer hat, ist die Injektion falscher Daten in die Berechnung (engl. *false data injection*). Diese zielt nicht auf die Vertraulichkeit, sondern auf die Korrektheit des

Protokolls ab, und ist in beiden seinen Varianten ($\pi_{\mathrm{IF}}$ und $\pi_{\mathrm{nIF}}$) trivial durchführbar. Der Angreifer, der nur einen Sensor korrumpiert hat, kann die Protokollausgaben beliebig (wenn das System beobachtbar ist, s. Seite 4) verfälschen, indem er seinen tatsächlich gemessenen Vektor $\hat{\underline{z}}_k^j$ durch einen beliebigen anderen ersetzt und die Kovarianz $\mathbf{C}_k^z$ so klein setzt, wie es die Quantisierung zulässt, wodurch die gefälschte Messung mit einem überproportional großem Gewicht in den Filterschritt einfließt. Dieser Angriff ist auch bei einer Klartextberechnung möglich, die Verschlüsselung macht aber die üblichen Plausibilitätsprüfungen [53, 59] unpraktikabel. In dieser Hinsicht ist es ein große Frage für die zukünftige Forschung, ob sich diese Verfahren gegen stärkere Angreifer sicher machen lassen oder ob „manche Funktionen an sich einfach unsicher sind“ [73].

Zum Schluss muss noch erwähnt werden, dass wir zwecks besserer Übersichtlichkeit auf die Nachrichtenauthentifizierung im Sensornetzwerk nicht näher eingegangen sind. Jede praktische Umsetzung, v. a. wenn über ungesicherte Kanäle kommuniziert wird, müsste die Authentizität der Nachrichten mit digitalen Signaturen bzw. MACs (engl. *message authentication codes* [48, Kap. 4]) sicherstellen, um die triviale *Man-in-the-Middle*-Angriffe vorzubeugen.

**Effizienz** Im Abschnitt 4.4 haben wir gezeigt, dass der zusätzliche Rechenaufwand, den die Sensoren zum Verschlüsseln und der Agent zum Entschlüsseln der Daten jede Kommunikationsrunde erbringen, quadratisch in der Zustandsvektorgröße $L$, aber konstant in der Netzwerkgröße $N$ ist. Der Zusatzaufwand, den die Hubs bei der homomorphen Aggregation der Messungen betreiben, ist $O(\sqrt{N})$, wenn die dreistufige Architektur mit $h$ Zwischenhubs nach der Formel (4.4) verwendet wird, bzw. $O(1)$ bei einer Binärbaumarchitektur. Der Gesamtnetzwerkaufwand ist $O(N)$ in der Setup-Phase (Schlüsselverteilung) und anschließend $O(N \cdot L^2)$ jede Kommunikationsrunde. Daraus schließen wir, dass das Protokoll $\pi_{\mathrm{IF}}$ (sowie $\pi_{\mathrm{nIF}}$, aus analogen Überlegungen) gut in der Netzwerkgröße skaliert und bei einem zeitinvarianten $L$ für Echtzeitapplikationen geeignet ist.

**Genauigkeit** Im Abschnitt 4.6 haben wir die Protokolle $\pi_{\mathrm{IF}}$ bzw. $\pi_{\mathrm{nIF}}$ numerisch simuliert und die Genauigkeit der Schätzungen evaluiert, die auf Basis deren Ausgaben erstellt wurden. Fest stand von Anfang an, dass die Ver- und Entschlüsselung der Daten keine Auswirkung auf die Genauigkeit des Verfahrens haben können, weil dies der Korrektheit des verwendeten Paillier-Schemas widersprechen würde. Die Hauptquelle des Fehlers (ggü. einer Berechnung des Verfahrens ohne Verschlüsselung und homomorphen Operationen) war deshalb die Quantisierung der Eingaben, die notwendig war, weil das Paillier-Schema keine Gleitkommaarithmetik unterstützt.

Unsere Evaluation hat gezeigt, dass bereits mit einer 16-Bit-Quantisierung (d. h. Rundung auf 16 Bitstellen nach dem Komma) die Genauigkeit der Schätzungen mittels unserer Protokolle mit der Filterung mittels Gleitkommaarithmetik im Klartext vergleichbar ist.

KAPITEL 5

# Sicheres Gossip-Konsensfilter

Im letzten Kapitel haben wir gezeigt, dass effiziente verteilte Messdatenfusion in der verschlüsselten Domäne möglich ist. Jedoch hat unser Informationsfilterprotokoll eine wichtige Engstelle bzgl. seiner Sicherheit und Skalierungsfähigkeit gehabt, nämlich die Aggregationshubs, die deshalb attraktive Angriffsziele sind. In diesem Kapitel bauen wir auf bisherigen Erkenntnissen auf und beantworten die Frage, ob vertrauliche verteilte Zustandsfilterung ohne eine zentrale Instanz möglich ist.

Im Abschnitt 2.1.3 haben wir mehrere Konsensfilter-Ansätze vorgestellt. Da das vollständige Kalman-Konsensfilter in der verschlüsselten Domäne auf die gleiche Problematik mit der Matrixinversion stößt, wie das klassische KF im Kapitel 3, fokussieren wir uns im Folgenden auf die Gossip-Konsensfilter. Im Abschnitt 5.1 stellen wir zwei Sicherheitsdefinitionen dafür auf: eine für ein Gossip-Filter, das analog unserem IF-Protokoll in der Fixkommaarithmetik homomorph berechnet wird, und eine, die die Eigenschaften der modularen Arithmetik ausnutzt, um in endlicher Rundenzahl die Konvergenz zu erreichen.

Im Abschnitt 5.2 beschreiben wir das Protokoll für das Gossip-Filter mit reellen bzw. quantisierten Eingaben und beweisen seine Sicherheit. Im Abschnitt 5.3 machen wir das Gleiche für das Gossip-Filter in endlichen Körpern, und analysieren beide bzgl. ihrer Komplexität im Abschnitt 5.4. Im Abschnitt 5.5 präsentieren wir die Ergebnisse der numerischen Evaluation einer prototypischen Umsetzung des ersten Protokolls, und im Abschnitt 5.6 bewerten wir beide in Bezug auf ihre Sicherheit, Skalierbarkeit und Genauigkeit.

## 5.1 Sicherheitsmodelle

Wir betrachten folgendes Szenario: Ein Sensornetzwerk überwacht einen skalaren Systemzustand, wobei jeder Sensor $P_j$ lokal das Kalman-Filter mit seinen eigenen Messungen laufen lässt und zu jedem Zeitpunkt $k$ seine aktuelle Zustandsschätzung

$\hat{x}_k^j$ (im Folgenden einfach mit $\hat{x}^j$ bezeichnet) kennt. Die Sensoren sind vernetzt, kommunizieren aber noch nicht untereinander. Eine externe Partei $P_{N+1}$ (Kontrolleur) möchte einen Konsenswert aus allen aktuellen Schätzungen bilden lassen, ihn auslesen, und an alle Sensoren verteilen.

**Definition 5.1 (Netzwerkarchitektur)** Wir stellen die Netzwerkarchitektur als einen ungerichteten Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ dar, wobei die Knoten $\mathcal{V}$ die $N$ Sensoren und die Kanten $\mathcal{E}$ die Verbindungen dazwischen repräsentieren. Wir definieren die Menge der Nachbarknoten von $j \in \mathcal{V}$ als $\mathcal{N}_j \equiv \{n : n \in \mathcal{V} \wedge (j, n) \in \mathcal{E}\}$.

Zusätzlich definieren wir die Gewichtungsmatrix $\mathbf{W} \in \mathbb{R}^{N \times N}$ als eine symmetrische zeilenstochastische Adjazenzmatrix von $\mathcal{G}$, für deren Elemente gilt: $w_{xy} > 0$, wenn $y \in \mathcal{N}_x \cup \{x\}$, ansonsten $w_{xy} = 0$.

Im Folgenden definieren wir zwei ideale Funktionalitäten: die Def. 5.2 repräsentiert das Gossip-Konsensfilter mit reellen Eingaben und die Def. 5.4, das Gossip-Konsensfilter in endlichen Körpern nach [65, 30] (s. auch die Diskussion auf Seite 9).

**Definition 5.2 (Ideales Gossip-Konsensfilter mit reellen Eingaben)** Wir definieren eine ideale Funktionalität zum Berechnen des Gossip-Konsensfilters mit $N$ reellen Eingaben und $N + 1$ Parteien wie folgt:

$$f_{\mathsf{GCR}}(E_1, E_2, \ldots, E_N, \lambda) = (\alpha, \alpha, \ldots, \alpha, \alpha),$$

wobei $N = \mathsf{poly}(n) \geq 2$, $T = \mathsf{poly}(n) \geq 1$ und $\lambda$ für eine leere Eingabe steht.

Für die Eingaben gilt $E_j = \left(\hat{x}^j, \underline{w}_j\right)$, wobei $\hat{x}^j \in \mathbb{R}$ die aktuelle Zustandsschätzung von $P_j$ und $\underline{w}_j$ die $j$-te Zeile der Gewichtungsmatrix $\mathbf{W}$ nach der Def. 5.1 ist. Für Ausgabe gilt:

$$\begin{aligned} \underline{\alpha} &= \mathbf{W}^{(T)} \begin{bmatrix} \hat{x}^1 & \hat{x}^2 & \ldots & \hat{x}^N \end{bmatrix}^{\mathrm{T}} \\ \alpha &\leftarrow \underline{\alpha}[i], \end{aligned}$$

wobei $\mathbf{W}^{(T)}$ für die $T$-fache Multiplikation von $\mathbf{W}$ mit sich selbst steht, $i$ unabhängig und gleichverteilt aus $\{1, \ldots, N\}$ gezogen wird, und $\underline{\alpha}[i]$ das $i$. Element des Vektors $\underline{\alpha}$ bezeichnet.

Die Def. 5.2 entspricht intuitiv der Ausführung des klassischen Gossipfilters mit $T$ Runden, denn die Potenzierung der Gewichtungsmatrix $\mathbf{W}$ ist nichts anderes, als die mehrfache Anwendung des Updateschritts nach der Formel (2.8). Weil wir im Allgemeinen nicht garantieren können, dass die Schätzungen nach $T$ Runden zu dem genauen Mittelwert konvergieren, gibt die Funktion stattdessen die Schätzung eines

zufällig ausgewählten Sensors $P_i$ aus, die erwartungsgemäß *nah* am Mittelwert liegen muss. Eine solche Formulierung der Funktionalität erspart uns die Notwendigkeit, die Fehlertoleranzen der finalen Schätzung explizit zu formalisieren.

Eine viel stärkere Konvergenzgarantie gibt uns das Gossip-Konsenfilter in endlichen Körpern [65], das eine vielversprechende Alternative für $f_{\mathrm{GCR}}$ sein kann, vorausgesetzt die modulare Gewichtungsmatrix $\mathbf{W}_s$ ist effizient berechenbar. Dieses Verfahren ist ein Beispiel für die Synergie zwischen den Feldern der Kryptographie und der Signalverarbeitung, in dem eine typische Signalverarbeitungsaufgabe (Mittelwertbildung) durch das Ausnutzen der Eigenschaften einer typischen kryptographischen Domäne (endliches Körper) gelöst wird.

**Definition 5.3** Wir bezeichnen die hypothetische Funktion, die zu jedem Netzwerkgraph $\mathcal{G}$ und Klartexmodulus $s$ eine Gewichtungsmatrix $\mathbf{W}_s \in \mathbb{Z}_s^{N\times N}$, die alle Bedingungen aus dem Abschnitt 2.1.3 erfüllt, sowie die Gossip-Rundenanzahl $T$ bis zur Konvergenz ausgibt, mit

$$FindModWeights\,(\mathcal{G}, s) = (\mathbf{W}_s, T)\,.$$

**Definition 5.4 (Ideales Gossip-Konsensfilter in endlichen Körpern)** Wir definieren eine ideale Funktionalität zum Berechnen des Gossip-Konsensfilters mit $N$ quantisierten Eingaben und $N+1$ Parteien wie folgt:

$$f_{\mathsf{GCZ}}(\hat{x}^1, \hat{x}^2, \ldots, \hat{x}^N, \mathcal{G}) = (A_1, A_2, \ldots, A_N, \alpha),$$

wobei $N = \mathsf{poly}(n) \geq 2$, $\lambda$ für eine leere Eingabe steht, $\hat{x}^j \in \mathbb{Z}_s$ die quantisierten Zustandsschätzung ist, der Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ die Sensornetzwerkarchitektur abbildet,

$$\alpha = \frac{1}{N}\sum_{j=1}^{N} \hat{x}^j \quad \text{und} \tag{5.1a}$$

$$A_j = \left(s, \underline{w}_j, T, \alpha\right), \tag{5.1b}$$

wobei $s$ das Klartextmodulus, $\underline{w}_j$ die $j$-te Zeile der Gewichtungsmatrix $\mathbf{W}_s \in \mathbb{Z}_s^{N\times N}$ und $\alpha \in \mathbb{R}$ ist.

Im Gegensatz zur Def. 5.2 hat der Kontrolleur $P_{N+1}$ hier von Anfang an die kompletten Graphinformationen, während die Sensorknoten nur ihre eigenen Schätzungen kennen und die Informationen über ihre Nachbarschaft und das Filterverfahren erst durch die Ausführung der Funktion erfahren. Dies ist notwendig, weil der Kontrolleur im realen Protokoll den öffentlichen Schlüssel bestimmt, aus dem sich das Klartextmodulus $s$ ergibt, von dem ihrerseits die Werte der modularen Gewichtungsmatrix $\mathbf{W}_s$ und die Gossip-Rundenanzahl $T$ abhängen. Durch das Aufnehmen dieser Werte

in die Ausgabe der Sensoren quantisieren wir formal die Informationen, die sie zur erfolgreichen Protokollausführung benötigen: $s$ hängt mit $pk$ zusammen, $\underline{w}_j$ brauchen sie im Klartext für die homomorphe Multiplikation, und $T$ müssen sie wissen, um den Updateschritt bis zur Konvergenz zu wiederholen.

Zu beachten ist außerdem, dass keine der beiden Funktionen vorgibt, ob und wie die Sensoren den Konsenswert $\alpha$ in ihre eigenen Schätzungen fusionieren. Ob er weiterverwendet oder verworfen wird, ist der konkreten Implementierung überlassen.

**Angreifermodell** Wie auch in den Kapiteln 3 und 4, nehmen wir einen statischen semiehrlichen (passiven) PPT-beschränkten (effizienten) Angreifer an.

## 5.2 Sicheres Gossip-Konsenfilter mit reellen Eingaben

### 5.2.1 Protokoll

Wir schlagen das Protokoll 5.1 ($\pi_{\mathrm{GCR}}$) auf der Seite 73 vor, um die ideale Funktionalität $f_{\mathrm{GCR}}$ sicher zu implementieren. Die Zustandsschätzungen $\hat{x}^j$ werden darin als $l_x$-Bit Integer mit $f_x < l_x$ Bits nach dem Komma quantisiert (s. Def. 2.10), und wir nehmen an, dass der Sicherheitsparameter $n$ groß genug ist, dass Integer der Bitlänge $l_x + T \cdot f_w$ ohne Überlauf verschlüsselt werden können.

Weil die stochastische Gewichtungsmatrix $\mathbf{W} \in \mathbb{R}^{N \times N}$ fürs Protokoll 5.1 auf $f_w$ Bitstellen nach dem Komma quantisiert wird (was die Voraussetzung für die homomorhe Multiplikation im Schritt 6 ist), addieren sich die Zeilen der quantisierten Matrix zu $2^{f_w}$ statt zu 1. Daraus folgt u. a., dass $\forall j \in \{1, \ldots, N\} : \underline{1}^{\mathrm{T}} \underline{w}_j = 2^{f_w}$, und davon leitet sich der Skalierungsfaktor im Schritt 10 ab.

Die Gossip-Rundenanzahl $T$ ist im $\pi_{\mathrm{GCR}}$ durch die Quantisierung und die maximale Klartextlänge in Bits (d. h. letztendlich durch die Größe des Klartextmodulus $s$) beschränkt, um modulare Überläufe zu vermeiden:

$$T \leq \left\lfloor \frac{\lceil \log_2 s \rceil - l_x}{f_w + 1} \right\rfloor. \tag{5.2}$$

### 5.2.2 Sicherheitsbeweis

Das Protokoll 5.1 ($\pi_{\mathrm{GCR}}$) ist im Gegensatz zum Protokoll 5.2 ($\pi_{\mathrm{GCZ}}$) *probabilistisch*, weil es keine Konvergenz garantieren kann und seine Gesamtausgabe deshalb zufällig aus den Ausgaben der einzelnen Sensoren gezogen wird. Aus diesem Grund reicht

**Protokoll 5.1 (Sicheres Gossip-Konsenfilter mit reellen Eingaben $\pi_{\mathrm{GCR}}$)**

- **Parameter**: Alle Parteien kennen $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$, die Netzwerkgröße $N$, die Gossip-Rundenanzahl $T$, die Quantisierungsstufe der Gewichte $f_w$, und den Sicherheitsparameter $n$.
- **Eingaben**:
  - Jede Partei $P_{j\in\{1,\dots,N\}}$ (Sensor) hat ihre quantisierte aktuelle Zustandsschätzung $\hat{x}^j$ und die $j$-te Zeile $\underline{w}_j$ der Gewichtungsmatrix $\mathbf{W}$ nach der Def. 5.1, quantisiert auf $f_w$ Bits nach dem Komma.
  - Die Partei $P_{N+1}$ (Kontrolleur) hat keine bzw. eine leere Eingabe $\lambda$.
- **Das Protokoll**:
  1. $P_{N+1}$ berechnet $(pk, sk) \leftarrow \mathsf{KeyGen}\,(1^n)$.
  2. $P_{N+1}$ zieht $i \leftarrow \{1, \dots, N\}$ und sendet $(pk, i)$ an $P_1, \dots, P_N$.
  3. Alle Parteien setzen eine interne Variable $t \coloneqq 1$.
  4. Jede $P_{j\in\{1,\dots,N\}}$ berechnet $c_t^j \leftarrow \mathsf{Enc}(pk, \hat{x}^j)$.
  5. Jede $P_{j\in\{1,\dots,N\}}$ sendet $c_t^j$ an ihre jeweiligen Nachbarn $P_{j\in\mathcal{N}_j}$.[a]
  6. Jede $P_{j\in\{1,\dots,N\}}$ berechnet für alle $n \in \mathcal{N}_j \cup \{j\}$
  $$d_t^n \leftarrow c_t^j \odot_{pk} \underline{w}_j[n].$$
  7. Jede $P_{j\in\{1,\dots,N\}}$ berechnet für alle $n_1, n_2, \dots \in \mathcal{N}_j$
  $$c_{t+1}^j \leftarrow d_t^j +_{pk} d_t^{n_1} +_{pk} d_t^{n_2} +_{pk} \dots$$
  8. Alle Parteien setzen $t \coloneqq t + 1$ und, wenn $t \leq T$, springen zum Schritt 5 zurück.
  9. $P_i$ sendet $c_{T+1}^i$ an $P_{N+1}$.
  10. $P_{N+1}$ berechnet $2^{-T\cdot f_w} \cdot \mathsf{Dec}(sk, c_{T+1}^i)$, um $\alpha \in \mathbb{R}$ zu erhalten, und sendet sie an $P_1, \dots, P_N$ im Klartext.
  11. Alle Parteien geben $\alpha$ aus.

a Per Def. 5.1 gilt $\mathcal{N}_j = \{n : \underline{w}_j[n] > 0 \wedge n \neq j\}$.

uns die einfache Sicherheitsformulierung nicht mehr aus, und wir müssen stattdessen die vollständige Formulierung nach der Def. 2.7 verwenden.

Die vollständige Formulierung verlangt, dass für jede Korruptionsstrategie (Untermenge $Bad \subset \{1, \ldots, N+1\}$ der Parteien, die der Angreifer korrumpieren kann), ein PPT-Algorithmus $\mathcal{S}_{Bad}$ existiert, so dass folgende Verteilungen ununterscheidbar sind:

$$\begin{aligned}&\left\{\mathcal{S}_{Bad}\left(1^n, \{E_j : j \in Bad\}, \alpha\right), f_{\mathrm{GCR}}(E_1, E_2, \ldots, E_N, \lambda)\right\} \stackrel{\mathrm{c}}{\equiv} \\ &\left\{\left\{\mathrm{view}_j^{\pi_{\mathrm{GCR}}}(E_1, E_2, \ldots, E_N, \lambda) : j \in Bad\right\}, \mathrm{output}^{\pi_{\mathrm{GCR}}}(E_1, E_2, \ldots, E_N, \lambda)\right\}. \quad (5.3)\end{aligned}$$

Dabei besteht die $\mathrm{view}^{\pi_{\mathrm{GCR}}}_{j\in\{1,\ldots,N\}}$ aus den Eingaben $\hat{x}^j, \underline{w}_j$, dem Zufallsband, und folgenden Nachrichten:

- $(pk, i)$ von $P_{N+1}$,
- $T \cdot |\mathcal{N}_j|$ Chiffraten von den Nachbarn des jeweiligen Sensors und
- $\alpha$ von $P_{N+1}$.

Die $\mathrm{view}^{\pi_{\mathrm{GCR}}}_{N+1}$ besteht nur aus den gemeinsamen Eingaben und einer Nachricht $c^i_{T+1} = [\![2^{T\cdot f_w}\alpha]\!]$ von $P_i$. Die reale Ausgabe $\mathrm{output}^{\pi_{\mathrm{GCR}}}(E_1, E_2, \ldots, E_N, \lambda)$ ist nach dem Lemma 5.1 eine $(N+1)$-fache Wiederholung des durch $P_{N+1}$ berechneten Werts $\alpha$. Im Folgenden kürzen wir $(\alpha, \alpha, \ldots, \alpha, \alpha)$ einfach mit „$\alpha$“ ab.

***Lemma 5.1*** *Unter der Annahme, dass* $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ *ein asymmetrisches Verschlüsselungsschema mit additivem Homomorphismus und perfekter Korrektheit ist, implementiert das Protokoll* $\pi_{\mathrm{GCR}}$ *die Funktionalität* $f_{\mathrm{GCR}}$ *korrekt, d. h. es gilt*

$$\{\mathrm{output}^{\pi_{\mathrm{GCR}}}(E_1, E_2, \ldots, E_N, \lambda)\}_{E_1,\ldots,E_N} \equiv \{(\alpha, \alpha, \ldots, \alpha, \alpha)\}_{E_1,\ldots,E_N},$$

*wobei* $\alpha$ *ein gleichverteilt gezogener Element des Vektors* $\underline{\alpha}$ *ist, mit*

$$\underline{\alpha} = \mathbf{W}^{(T)} \begin{bmatrix}\hat{x}^1 & \hat{x}^2 & \ldots & \hat{x}^N\end{bmatrix}^{\mathrm{T}}.$$

Beweis. Zuerst beobachten wir, dass der Wert $\alpha$, den alle Parteien im letzten Schritt des Protokolls $\pi_{\mathrm{GCR}}$ ausgeben, im Schritt 10 vom Kontrolleur durch

$$\alpha = 2^{-T\cdot f_w} \cdot \mathsf{Dec}(sk, c^i_{T+1})$$

berechnet und an alle versendet wird. $i$ wird vom Kontrolleur in der Setup-Phase gleichverteilt gezogen und $c^i_{T+1}$ enthält die $T$-te fusionierte Zustandsschätzung des Sensors $P_i$. Dadurch bleibt uns nur noch zu zeigen, dass folgende Gleichheit gilt:

$$\begin{bmatrix}\mathsf{Dec}(sk, c^1_{T+1}) \\ \vdots \\ \mathsf{Dec}(sk, c^N_{T+1})\end{bmatrix} = 2^{T\cdot f_w}\mathbf{W}^{(T)} \begin{bmatrix}\hat{x}^1 \\ \vdots \\ \hat{x}^N\end{bmatrix}.$$

Der Term $2^{T \cdot f_w}\mathbf{W}^{(T)}$ lässt sich auch als $\left(2^{f_w}\mathbf{W}\right)^{(T)}$ umschreiben, und $2^{f_w}\mathbf{W}$ ist gerade die auf $f_w$ Bits nach dem Komma quantisierte Gewichtungsmatrix, die wir im Folgenen mit $\mathbf{W}_q$ bezeichnen.

Im Schritt 6 gilt für alle $n \in \{1, \ldots, N\} \setminus (\mathcal{N}_j \cup \{j\})$ per Def. 5.1 $\underline{w}_j[n] = 0$. Damit ist es leicht einsehbar, dass die Schritte 6 und 7 eigentlich nur eine homomorphe Vektor-Matrix-Multiplikation darstellen, konkret:

$$\begin{bmatrix} \mathsf{Dec}(sk, c_{t+1}^1) \\ \vdots \\ \mathsf{Dec}(sk, c_{t+1}^N) \end{bmatrix} = \mathbf{W}_q \begin{bmatrix} \mathsf{Dec}(sk, c_t^1) \\ \vdots \\ \mathsf{Dec}(sk, c_t^N) \end{bmatrix}$$

Wir wissen, dass im Schritt 4 $c_1^j = \mathsf{Enc}(pk, \hat{x}^j)$ ist und dass die Multiplikation $T$-mal durchgeführt wird. Damit gilt letztendlich

$$\begin{bmatrix} \mathsf{Dec}(sk, c_{T+1}^1) \\ \vdots \\ \mathsf{Dec}(sk, c_{T+1}^N) \end{bmatrix} = \mathbf{W}_q^{(T)} \begin{bmatrix} \mathsf{Dec}(sk, c_1^1) \\ \vdots \\ \mathsf{Dec}(sk, c_1^N) \end{bmatrix} = \left(2^{f_w}\mathbf{W}\right)^{(T)} \begin{bmatrix} \hat{x}^1 \\ \vdots \\ \hat{x}^N \end{bmatrix} = 2^{T \cdot f_w}\mathbf{W}^{(T)} \begin{bmatrix} \hat{x}^1 \\ \vdots \\ \hat{x}^N \end{bmatrix}.$$

Weil der vom Protokoll $\pi_{\mathrm{GCR}}$ und von der idealen Funktionalität $f_{\mathrm{GCR}}$ berechnete Vektor $\underline{\alpha}$ genau gleich ist, und weil beide ihre Ausgaben daraus unabhängig und gleichverteilt ziehen, sind ihre jeweiligen Ausgaben identisch verteilt. □

***Satz 5.2*** *Unter den Annahmen, dass a)* (KeyGen, Enc, Dec, Eval) *ein IND-CPA sicheres asymmetrisches Verschlüsselungsschema mit komplexitätstheoretisch starkem additivem Homomorphismus ist, und dass b) kein Angreifer gleichzeitig $P_{N+1}$ und eine beliebige andere Partei korrumpieren kann, berechnet $\pi_{\mathrm{GCR}}$ die Funktionalität $f_{\mathrm{GCR}}$ in der Anwesenheit der statischen semiehrlichen PPT-Angreifern sicher.*

Beweis. Zuerst betrachten wir die möglichen Einzelkorruptionsfälle.

**Korruption eines Sensors** Für den Fall, dass der Sensor $P_1$ korrumpiert ist, konstruieren wir einen Simulator $\mathcal{S}_1$, der die Eingaben $1^n, \hat{x}^1, \underline{w}_1, \alpha$ bekommt und wie folgt vorgeht:

1. $\mathcal{S}_1$ zieht für $P_4$ ein uniform verteiltes Zufallsband $r^1$ der ausreichenden Länge.
2. $\mathcal{S}_1$ berechnet $(pk, sk) \leftarrow \mathsf{KeyGen}\,(1^n)$ mit unabhängigem Zufall.
3. $\mathcal{S}_1$ zieht $i \leftarrow \{1, \ldots, N\}$ mit unabhängigem Zufall.
4. $\mathcal{S}_1$ führt $\mathsf{Enc}(pk, 0)$ insgesamt $T\,|\mathcal{N}_1|$ Mal mit unabhängigem Zufall aus, um die Chiffrate $u_1, \ldots, u_{T|\mathcal{N}_j|}$ zu erhalten.
5. $\mathcal{S}_1$ gibt aus: $\left(\hat{x}^1, \underline{w}_1, r^1; pk, i, u_1, \ldots, u_{T|\mathcal{N}_1|}, \alpha\right)$.

Um zu zeigen, dass die Verbundverteilung $\{\mathcal{S}_1(1^n, (\hat{x}^1, \underline{w}_1), \alpha), \alpha\}$ nicht von der realen Verteilung $\{\text{view}_1^{\pi_{\text{GCR}}}, \text{output}^{\pi_{\text{GCR}}}\}$ unterscheidbar ist, sehen wir zuerst ein, dass der Wert $\alpha$ nach dem Lemma 5.1 im Realen und im Idealen identisch verteilt ist. Des Weiteren, weil $\mathcal{S}_1$ ihn als Eingabe bekommt und unverändert ausgibt, ist die Simulatorausgabe automatisch mit der Gesamtausgabe konsistent. $pk$ und $i$ werden vom Simulator genauso erzeugt, wie es eine ehrliche $P_{N+1}$ tun würde, deshalb sind sie auch identisch verteilt.

Die Chiffrate $u_1, \ldots, u_{T|\mathcal{N}_1|}$ sind von den echten Nachrichten der Nachbarn komplexitätstheoretisch ununterscheidbar, denn wir können analog dem Beweis 4.2 ein Hybridargument aufstellen, dass eine Sequenz „frischer“ Nullchiffrate unter der Annahme a) nicht von einer gleich langen Sequenz homomorph evaluierter Chiffrate unterscheidbar ist. Des Weiteren behaupten wir, dass die gefälschten Chiffrate immer mit dem gefälschtem Index $i$ konsistent sind, denn das Chiffrat $c_{T+1}^i$, aus dem $P_{N+1}$ den Wert $\alpha$ berechnet, bekommt $P_1$ nie zugesendet. Und selbst wenn $i = 1$ und $P_1$ das $c_{T+1}^1$ eigenständig aus den Chiffraten $c_T^{1,\text{w}} \cup \{c_T^{n,\text{w}} : n \in \mathcal{N}_1\}$ homomorph erzeugt, kann kein PPT-Unterscheider erkennen, ob $c_{T+1}^1$ tatsächlich den Klartext $2^{T \cdot f_w}\alpha$ enthält oder nicht, denn das würde der Annahme a) widersprechen.

Für $\mathcal{S}_2, \ldots, \mathcal{S}_N$ erfolgt die Konstruktion und die Beweisführung analog.

**Korruption des Kontrolleurs** Nun betrachten wir den Fall, dass der Kontrolleur $P_{N+1}$ korrumpiert ist, und konstruieren dafür einen Simulator $\mathcal{S}_{N+1}$, der die Eingaben $1^n, \alpha$ bekommt und wie folgt vorgeht:

1. $\mathcal{S}_{N+1}$ zieht für $P_{N+1}$ ein uniform verteiltes Zufallsband $r^{N+1}$ der ausreichenden Länge.
2. $\mathcal{S}_{N+1}$ berechnet $(pk, sk) \leftarrow \mathsf{KeyGen}(1^n)$ mit $r^{N+1}$.
3. $\mathcal{S}_{N+1}$ berechnet $v \leftarrow \mathsf{Enc}(pk, 2^{T \cdot f_w}\alpha)$ mit unabhängigem Zufall.
4. $\mathcal{S}_{N+1}$ gibt aus: $\left(r^{N+1}; v\right)$.

Um zu zeigen, dass die Verbundverteilung $\{\mathcal{S}_{N+1}(1^n, \alpha), \alpha\}$ nicht von der realen Verteilung $\left\{\text{view}_{N+1}^{\pi_{\text{GCR}}}, \text{output}^{\pi_{\text{GCR}}}\right\}$ unterscheidbar ist, müssen wir nur überprüfen, ob $v$ mit $\alpha$ konsistent ist, denn gegeben $r^{N+1}$ kann jeder Unterscheider effizient den $sk$ nachrechnen und $v$ wieder entschlüsseln. Diese Konsistenz folgt aber direkt daraus, dass der verschlüsselte Klartext im Schritt 3 des Simulators die Umkehrung der Division im Schritt 11 des Protokolls ist, und dass per Annahme a) ein frisches Chiffrat von einem evaluierten mit dem gleichen Inhalt komplexitätstheoretisch ununterscheidbar ist.

An dieser Stelle ist es für die formale Sicherheit wichtig, dass die Ausgabe $\alpha \in \mathbb{R}$ bleibt und nicht auf $l_x$ Bit quantisiert wird, denn die Quantisierung die $T \cdot f_w$ niedrigstwertigen Bits verwerfen würde, anhand deren ein Unterscheider die Fälschung durch den Simulator trivial erkennen würde. Für die Sicherheit der realen Applikationen, die ihre Variablen natürlich quantisieren müssen, hätte dies die Implikation, dass der Kontrolleur „zu viele“ niedrigstwertige Bits des Konsensmittelwerts lernt. Das Informationsgehalt dieser Bits ist aber i. d. R. mit dem Messrauschen vergleichbar, deshalb betrachten wir die reale Gefahr dieser „Datenpanne“ als minimal.

**Mehrfachkorruption** Als letztes betrachten wir die Mehrfachkorruptionsszenarien. Per Annahme b) lassen wir dabei die gleichzeitige Korruption des Kontrolleurs und eines oder mehrerer Sensoren außen vor. Diese Annahme benötigen wir für den Sicherheitsbeweis, weil im benannten Fall der Angreifer den Geheimschlüssel des Kontrolleurs benutzen könnte, um die Nachrichten zu entschlüsseln, die die ehrlichen Parteien an ihre korrumpierten Nachbarn senden. Damit käme er an die Daten ran, die ein Simulator in der idealen Welt nur mit einer in $l_x$ vernachlässigbaren Wahrscheinlichkeit simulieren (d. h. raten) könnte.

Damit bleibt noch ein Szenario, das wir betrachten müssen, nämlich die gleichzeitige Korruption mehrerer Sensoren. Bezeichnen wir die Menge der Indizies dieser Sensoren mit $Bad \subset \{1, \ldots, N\}$, dann bekommt der Simulator $\mathcal{S}_{j \in Bad}$ (abgekürzt $\mathcal{S}_{Bad}$) die Eingaben $\left\{1^n, \hat{x}^j, \underline{w}_j, \alpha : j \in Bad\right\}$ und geht wie folgt vor:

1. $\mathcal{S}_{Bad}$ zieht für jede $P_{j \in Bad}$ ein uniform verteiltes Zufallsband $r^j$ der ausreichenden Länge.
2. $\mathcal{S}_{Bad}$ berechnet $(pk, sk) \leftarrow \mathsf{KeyGen}\,(1^n)$ mit unabhängigem Zufall.
3. $\mathcal{S}_{Bad}$ zieht $i \leftarrow \{1, \ldots, N\}$ mit unabhängigem Zufall.
4. $\mathcal{S}_{Bad}$ berechnet für $j \in Bad$ jeweils $u_1^j \leftarrow \mathsf{Enc}(pk, \hat{x}^j)$ mit $r^j$.
5. Für $t \in \{1, \ldots, T-1\}$:
   a) $\mathcal{S}_{Bad}$ berechnet für $n \in \bigcup_{j \in Bad} \mathcal{N}_j \setminus Bad$ jeweils $u_t^n \leftarrow \mathsf{Enc}(pk, 0)$ mit unabhängigem Zufall.
   b) $\mathcal{S}_{Bad}$ berechnet für $j \in Bad$ und jeweils $n \in \mathcal{N}_j \cup \{j\}$ sowie $n_1, n_2, \ldots \in \mathcal{N}_j$:

$$u_t^{n,\mathrm{w}} \leftarrow u_t^j \odot_{pk} \underline{w}_j[n]$$
$$u_{t+1}^j \leftarrow u_t^{j,\mathrm{w}} +_{pk} u_t^{n_1,\mathrm{w}} +_{pk} u_t^{n_2,\mathrm{w}} +_{pk} \ldots$$

6. $\mathcal{S}_{Bad}$ gibt für jede $P_{j \in Bad}$ aus: $\left(\hat{x}^j, \underline{w}_j, r^j; pk, i, \left\{u_1^{j \in \mathcal{N}_j}\right\}, \ldots, \left\{u_T^{j \in \mathcal{N}_j}\right\}, \alpha\right)$.

Intuitiv erreicht $\mathcal{S}_{Bad}$ die Ununterschedbarkeit seiner Ausgaben von dem realen Protokoll dadurch, dass er die gleiche homomorphe Evaluationen auf den Chiffraten ausführt, die auch die ehrlichen Parteien anwenden würden. Dafür verschlüsselt er die ihm bekannten Eingaben der korrumpierten Parteien mit ihrem Zufall bzw. erzeugt frische Nullchiffrate anstelle der Nachrichten ihrer ehrlichen Nachbarn. Weiter oben haben wir gezeigt, dass diese Nullchiffrate von den echten evaluierten Chiffraten ununterscheidbar sind, also sind auch die komplette simulierte Sichten vom realen Protokoll nicht unterscheidbar. Aus der gleichen Überlegung wie bei der Einzelsensorkorruption sind alle simulierten Nachrichten auch mit der Ausgabe $\alpha$ konsistent.

Wir haben nun für jede per Sicherheitsdefinition erlaubte Korruptionsstrategie einen Simulator konstruiert und gezeigt dass für ihre Ausgaben die Ununterscheidbarkeitsrelation (5.3) gilt. Dadurch haben wir bewiesen, dass das Protokoll $\pi_{\mathrm{GCR}}$ die Funktionalität $f_{\mathrm{GCR}}$ in der Anwesenheit der statischen semiehrlichen PPT-Angreifern sicher berechnet. □

## 5.3 Sicheres Gossip-Konsenfilter in endlichen Körpern

Das Protokoll $\pi_{\mathrm{GCR}}$ gibt uns keine theoretischen Garantien, dass die lokale Schätzungen einzelner Sensoren zu einem gemeinsamen Wert am Ende seiner Ausführung konvergieren. Wir können auch die Anzahl $T$ der Gossip-Runden nicht frei einstellen, weil ihr maximaler Wert gemäß der Gleichung (5.2) durch die Schlüssellänge und die Quantisierungsstufen der Schätzungen und der Gewichte begrenzt ist. Wird eine solche Konvergenzgarantie gefordert, liefert stattdessen der Gossip-Konsensfilter in endlichen Körpern einen Lösungsansatz.

### 5.3.1 Protokoll

Wir schlagen das Protokoll 5.2 ($\pi_{\mathrm{GCZ}}$) auf der Seite 79 vor, um die ideale Funktionalität $f_{\mathrm{GCZ}}$ sicher zu implementieren. Auch hier werden die Zustandsschätzungen $\hat{x}^j$ als $l_x$-Bit Integer mit $f_x < l_x$ Bits nach dem Komma quantisiert. Wir formalisieren die Annahme, dass die Funktion $FindModWeights$ nach der Def. 5.3 effizient berechenbar ist, indem wir den Protokollparteien Zugriff auf das gleichnamige Orakel geben. Die Implikationen davon für Sicherheitsbeweise gegen stärkere Angreifer, die vom Protokoll abweichen und mithilfe des Orakels ggf. auch andere $\mathcal{NP}$-harte Probleme lösen könnten, sind ein Thema für die zukünftige Recherche.

**Protokoll 5.2 (Sicheres Gossip-Konsenfilter in endlichen Körpern $\pi_{\mathrm{GCZ}}$)**

- **Parameter**: Alle Parteien kennen $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$, die Netzwerkgröße $N$, und den Sicherheitsparameter $n$.
- **Eingaben**:
  - Jede Partei $P_{j\in\{1,\dots,N\}}$ (Sensor) hat ihre quantisierte aktuelle Zustandsschätzung $\hat{x}^j$.
  - Die Partei $P_{N+1}$ (Kontrolleur) hat die Netzwerkinformation $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ nach der Def. 5.1.
- **Orakel**: Alle Parteien haben Zugriff auf das Orakel $FindModWeights$ nach der Def. 5.3.
- **Das Protokoll**:
  1. $P_{N+1}$ berechnet $(pk, sk) \leftarrow \mathsf{KeyGen}\,(1^n)$ und den semiprimen Klartextmodulus $s$ dazu.
  2. $P_{N+1}$ ruft das Orakel auf: $(\mathbf{W}_s, T) \coloneqq FindModWeights\,(\mathcal{G}, s)$.
  3. $P_{N+1}$ sendet an jede $P_{j\in\{1,\dots,N\}}$ jeweils $\big(pk, s, \underline{w}_j, T\big)$, wobei $\underline{w}_j \in \mathbb{Z}_s^N$ die $j$-te Zeile der modularen Gewichtungsmatrix $\mathbf{W}_s$ ist.
  4. Alle Parteien setzen eine interne Variable $t \coloneqq 1$.
  5. Jede $P_{j\in\{1,\dots,N\}}$ berechnet $c_t^j \leftarrow \mathsf{Enc}(pk, \hat{x}^j)$.
  6. Jede $P_{j\in\{1,\dots,N\}}$ sendet $c_t^j$ an ihre jeweiligen Nachbarn $P_{j\in\mathcal{N}_j}$.
  7. Jede $P_{j\in\{1,\dots,N\}}$ berechnet für alle $n \in \mathcal{N}_j \cup \{j\}$
  $$c_t^{n,\mathsf{w}} \leftarrow c_t^j \odot_{pk} \underline{w}_j[n].$$
  8. Jede $P_{j\in\{1,\dots,N\}}$ berechnet für alle $n_1, n_2, \dots \in \mathcal{N}_j$
  $$c_{t+1}^j \leftarrow c_t^{j,\mathsf{w}} +_{pk} c_t^{n_1,\mathsf{w}} +_{pk} c_t^{n_2,\mathsf{w}} +_{pk} \dots$$
  9. Alle Parteien setzen $t \coloneqq t + 1$ und, wenn $t \leq T$, springen zum Schritt 6 zurück.
  10. $P_1$ sendet $c_{T+1}^1$ an $P_{N+1}$.
  11. $P_{N+1}$ berechnet $(N \cdot \mathsf{Dec}(sk, c_{T+1}^1) \mod s)/N$, um $\alpha \in \mathbb{R}$ zu erhalten, sendet diese an $P_1, \dots, P_N$, und gibt sie aus.
  12. Jede $P_{j\in\{1,\dots,N\}}$ gibt aus: $\big(s, \underline{w}_j, T, \alpha\big)$.

### 5.3.2 Korrektheit

Grundsätzlich verläuft der Sicherheitsbeweis für das $\pi_{\mathrm{GCZ}}$ analog dem für das $\pi_{\mathrm{GCR}}$ 5.2. Weil das erstere aber im Gegensatz zu dem letzteren deterministisch ist, reicht uns dafür die einfachere Sicherheitsformulierung nach der Def. 2.8 aus.

***Satz 5.3*** *Unter der Annahme, dass* $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ *ein asymmetrisches Verschlüsselungsschema mit additivem Homomorphismus in* $\mathbb{Z}_s$ *und perfekter Korrektheit ist, implementiert das Protokoll* $\pi_{\mathrm{GCZ}}$ *die Funktionalität* $f_{\mathrm{GCZ}}$ *korrekt, d. h. es gilt*

$$\mathrm{output}^{\pi_{\mathrm{GCZ}}}(\hat{x}^1, \hat{x}^2, \ldots, \hat{x}^N, \mathcal{G}) = (A_1, A_2, \ldots, A_N, \alpha),$$

*wobei für* $\alpha$ *die Gleichung* (5.1a) *und für* $A_j = \big(s, \underline{w}_j, T, \alpha\big)$*, die Gleichung* (5.1b) *aus der Def. 5.4 gelten.*

Beweis. Die Ausgaben $s, \underline{w}_j, T$ der Parteien $P_{j\in\{1,\ldots,N\}}$ sind trivialerweise korrekt, denn sie werden von $P_{N+1}$ erzeugt bzw. vom Orakel $FindModWeights$ berechnet und von $P_j$ unverändert ausgegeben. $\alpha$ ist in der Ausgabe jeder Partei per Protokolldefinition gleich, und um zu zeigen, dass $\alpha$ der Mittelwert von $\hat{x}^1, \ldots, \hat{x}^N$ ist, müssen wir das Protokoll 5.2 nur auf den Algorithmus 1 in [65] reduzieren.

Diese Reduktion ist überschaubar, denn der Algorithmus 1 besteht im wesentlichen aus der $T$-fachen Wiederholung der drei Schritte, die auch in unserem Protokoll $T$-mal in den Schritten 6 bis 8 homomorph auf die Chiffrate angewendet werden: Austausch der aktuellen Schätzungen mit den Nachbarn, Gewichtung der eigenen und der empfangenen Schätzungen, und ihre additive Fusion. Dabei werden in beiden Fällen die Additionen und die Multiplikationen in einem endlichen Körper berechnet, für den bestimmte Vorbedingungen gelten müssen (s. Seite 9).

Die Ausgabe $\mathrm{mod}(N\hat{x}_{T+1}, s)\,/N$ wird in [65] von jedem Sensor lokal berechnet, weil aber die reelle Division nicht homomorph berechenbar ist, wird im $\pi_{\mathrm{GCZ}}$ stattdessen das Chiffrat $c^1_{T+1}$ an den Kontrolleur $P_{N+1}$ gesendet, der es entschlüsselt und die Ausgabe nach Vorschrift berechnet. Dabei spielt es keine Rolle, ob der Kontrolleur das Chiffrat von $P_1$ oder beliebiger anderer Partei bekommt, weil alle Schätzungen nach $T$ Runden garantiert zu einem Konsenswert konvergieren.

Damit haben wir gezeigt, dass das Protokoll $\pi_{\mathrm{GCZ}}$ den [65, Alg. 1] umsetzt, und weil der letztere den Mittelwert von $\hat{x}^1, \ldots, \hat{x}^N$ korrekt berechnet, implementiert auch unser Protokoll die ideale Funktionalität $f_{\mathrm{GCZ}}$ korrekt. □

### 5.3.3 Vertraulichkeit

***Satz 5.4*** *Unter den Annahmen, dass*

*a)* (KeyGen, Enc, Dec, Eval) *ein IND-CPA sicheres asymmetrisches Verschlüsselungsschema mit komplexitätstheoretisch starkem additivem Homomorphismus in $\mathbb{Z}_s$ ist,*

*b) kein Angreifer gleichzeitig $P_{N+1}$ und eine beliebige andere Partei korrumpieren kann, und*

*c) aus einem Klartextmodulus $s$ effizient ein gültiger öffentlicher Schlüssel $pk$ erzeugt werden kann, der von jedem anderen Schlüssel mit dem gleichen Klartextmodulus komplexitätstheoretisch ununterscheidbar ist,*

*berechnet $\pi_{\mathrm{GCZ}}$ die Funktionalität $f_{\mathrm{GCZ}}$ in der Anwesenheit der statischen semiehrlichen PPT-Angreifern sicher.*

Beweisskizze. Der Beweis wird analog dem für den Satz 5.2 geführt, allerdings mit etwas angepassten Ein- und Ausgaben der Sensorparteien. Während im $\pi_{\mathrm{GCR}}$ jeder Sensor die Eingaben $\big(\hat{x}^j, \underline{w}_j\big)$ und die Ausgabe $\alpha$ hatte, haben die Sensoren in $\pi_{\mathrm{GCZ}}$ nur $\hat{x}^j$ als Eingabe, geben aber $\big(s, \underline{w}_j, T, \alpha\big)$ aus. Für die Simulatoren ist es aber irrelevant, weil sie beides als Eingabe bekommen und unter der einfachen Sicherheitsformulierung die Ausgaben der ehrlichen Parteien nicht beachten müssen.

Die Nachrichten, die die Sensoren im jeweiligen Protokoll empfangen, unterscheiden sich nur darin, was $P_{N+1}$ an die Sensoren in der Setup-Phase verteilt. Seine Nachricht $\big(pk, s, \underline{w}_j, T\big)$ ist aber trivial simulierbar, denn drei Elemente davon sind bereits in der Ausgabe von $P_j$ enthalten, und ein vom echten Schlüssel ununterscheidbarer $pk$ lässt sich aus $s$ per Annahme c) effizient erzeugen. Ob das Paillier-Schema oder irgendein anderes homomorphes Schema diese Annahme erfüllt, ist allerdings unklar und wird ein Thema für die zukünftige Recherche sein.[1]

Mit dieser Beweisskizze haben wir gezeigt, dass das Protokoll $\pi_{\mathrm{GCZ}}$ die Funktionalität $f_{\mathrm{GCZ}}$ in der Anwesenheit der statischen semiehrlichen PPT-Angreifern unter den besagten Annahmen vertraulich berechnet. Daraus und aus dem Satz 5.3 folgt letztendlich, dass das Protokoll ggü. ebensolchen Angreifern sicher ist. □

## 5.4 Komplexitätsanalyse

Wir betrachten nun den Rechen- und Netzwerkaufwand der Protokolle 5.1 und 5.2 und bezeichnen im Folgenden die durchschnittliche Nachbarzahl eines Netzwerkknoten (den durchschnittlichen Knotengrad) mit $\hat{\mathcal{N}} = \frac{1}{N}\sum_{j=1}^{N} |\mathcal{N}_j|$.

1 Im Spezialfall des Paillier-Schemas lautet die Fragestellung: Kann ein solches $g \in \mathbb{Z}_{s^2}$ effizient gefunden werden, gegeben nur $s = pq$ (aber nicht $p, q$), dass die notwendige Bedingung $\gcd\big(L_n(g^\nu \bmod s^2), s\big) = 1$ per Def. 2.6 mit einer überwältigenden Wahrscheinlichkeit erfüllt ist?

**Rechenaufwand** Der Rechenaufwand eines Sensors besteht in beiden Protokollen aus einem einmaligen Aufruf von Enc, um die Schätzung $\hat{x}^j$ zu verschlüsseln, und $(\hat{\mathcal{N}}+1)$ Eval$(\times)$-Aufrufe sowie $\hat{\mathcal{N}}$ Eval$(+)$-Aufrufe pro Gossip-Runde. Wenn der Systemzustand durch einen Vektor der Länge $L$ (statt einem Skalar) repräsentiert wird, wachsen diese Zahlen um Faktor $L$, also insgesamt $L$ Enc- und $LT\left(2\hat{\mathcal{N}}+1\right)$ Eval-Aufrufe über die komplette Ausführung. Der Rechenaufwand pro Sensor ist linear in allen drei Hauptparametern $L, T, \hat{\mathcal{N}}$ und konstant in der Netzwerkgröße $N$.

Der Kontrolleur hat in beiden Protokollen den einmaligen Rechenaufwand zum Erzeugen eines Schlüsselpaars mit KeyGen und der $L$-fachen Ausführung von Dec zum Entschlüsseln des Konsensvektors, d. h. der Gesamtrechenaufwand ist linear in $L$ und konstant in $N$. Im $\pi_{\mathrm{GCZ}}$ kommt aber zusätzlich der Aufwand zum Berechnen von $FindModWeights\,(\mathcal{G}, s)$ hinzu, der im allgemeinen exponentiell in der Größe von $\mathcal{G}$ (also in $N$) ist [65].

**Netzwerkaufwand** Der Kontrolleur sendet in den Setup-Phasen beider Protokolle eine Nachricht an jeden Sensor, die seinen öffentlichen Schlüssel $pk$ enthält. Im $\pi_{\mathrm{GCZ}}$ enthält sie aber zusätzlich den Vektor $\underline{w}_j$ der Länge $N$, d. h. der Gesamtnetzwerkaufwand in der Setup-Phase ist zuerst $O(N^2)$, lässt sich aber auf $O(N \cdot \hat{\mathcal{N}})$ reduzieren, wenn nur die jeweiligen Nachbargewichte tatsächlich übertragen werden.

Die Sensoren senden pro Gossip-Runde $L \cdot \hat{\mathcal{N}}$ verschlüsselte Nachrichten an ihre Nachbarn. Der Netzwerkaufwand über die komplette Ausführung ist also $O(TL\hat{\mathcal{N}})$ pro Sensor bzw. $O(NTL\hat{\mathcal{N}})$ im gesamten Netzwerk, also linear in jedem Parameter.

In der Abschlussphase sendet der Kontrolleur in beiden Protokolle $N$ Nachrichten an alle Sensoren, die den Mittelwert $\alpha$ enthalten.

## 5.5 Genauigkeitsauswertung mittels numerischer Simulation

Wie auch für das Protokoll $\pi_{\mathrm{IF}}$, haben wir für das $\pi_{\mathrm{GCR}}$ eine einfache Softwaresimulation in *Python* entwickelt, um die Genauigkeit des Verfahrens zu evaluieren. Ihr Quellcode ist öffentlich unter `https://github.com/KIT-ISAS/PP-GCF` zugänglich. Für das $\pi_{\mathrm{GCZ}}$ haben wir keine äquivalente Simulation umsetzen können, weil die praktische Umsetzung der Funktion $FindModWeights$ nach der Def. 5.3 außerhalb des Umfangs dieser Arbeit war; eine numerische Evaluation des unverschlüsselten Gossip-Konsensfilters in endlichen Körper ist aber in [65] enthalten.

**Simuliertes Szenario** Für die Evaluation des Gossip-Konsensfilters haben wir ein stark vereinfachtes Szenario entworfen. In unserer Simulation wird ein System si-

muliert, dessen interne Zustand (z. B. seine Temperatur) sich komplett mit einer skalaren Zufallsvariable $\boldsymbol{x}_k$ repräsentieren lässt ($L = 1$). Das System startet im Zustand $\overset{\star}{x}_0$ und verändert sich jede Zeitrunde $k \in \{1, \ldots, K\}$ um einen zufälligen Wert aus $\mathcal{N}(0, \sigma_w^2)$, d. h. das „Systemmodell“ ist $\mathbf{A} = [1], \mathbf{C}^w = [\sigma_w^2]$. In unserem simulierten Szenario waren $K = 50$, $\overset{\star}{x}_0 = 100\,°\mathrm{C}$ und $\sigma_x = 2.5\,°\mathrm{C}$ eingestellt.

Das Sensornetzwerk besteht aus $N = 64$ Knoten, die in einem $8 \times 8$ Raster angeordnet und mit ihren jeweils horizontalen, vertikalen und diagonalen Nachbarn verknüpft sind. Die Sensoren verfälschen den „erfassten“ Systemzustand $\overset{\star}{z}_k^i$ mit unabhängigem additivem Rauschen aus $\mathcal{N}(0, \sigma_z^2)$, um das Messmodell $\mathbf{H} = [1], \mathbf{C}^z = [\sigma_z^2]$ zu simulieren. Sie führen in unserer Simulation kein lokales Filtern aus und speichern nur ihre neuesten Messungen.

Jede Zeitrunde $k$ setzen alle Sensoren $\hat{x}_{k,1}^j := \overset{\star}{z}_k^i$ und führen damit das Protokoll $\pi_{\mathrm{GCR}}$ mit $T$ Gossip-Runden aus, um jeweils die Konsensschätzung $\hat{x}_{k,T+1}^j$ zu erhalten. Die Nachbargewichte $\underline{w}_j$ werden dabei wie folgt berechnet:

$$\forall n \in \{1, \ldots, N\} : \underline{w}_j[n] = \begin{cases} w_{jj}, & \text{wenn } n = j \\ \frac{1}{|\mathcal{N}_j|}\left(1 - w_{jj}\right), & \text{wenn } n \in \mathcal{N}_j \\ 0, & \text{sonst} \end{cases},$$

wobei der Gewicht der eigenen Schätzung für alle Sensoren auf $w_{jj} = 0.2$ eingestellt war. Es ist leicht zu überprüfen, dass die Matrix $\mathbf{W}$ dann zeilenstochastisch ist. Der Kontrolleur wählt am Anfang der Zeitrunde einen Sensor $P_i$ unabhängig und gleichverteilt aus $\{P_1, \ldots, P_N\}$ und erhält am Schluss seine Schätzung $\hat{x}_{k,T+1}^i$.

Der Parameter $T$ wurde nach der Formel (5.2) mit Klartextmoduluslänge von 192 Bit[2], $l_x = 32\,\mathrm{Bit}$ und $f_w = 7\,\mathrm{Bit}$ berechnet:

$$T = \left\lfloor \frac{192 - 32}{7 + 1} \right\rfloor = 20.$$

**Simulationsergebnisse** Analog der Evaluation von $\pi_{\mathrm{IF}}$ haben wir unser Verfahren mit unterschiedlichen Quantisierungsstufen $f_x$ in drei Szenarien mit jeweils gleichen System- und Messdaten evaluiert, wobei die Szenarien sich nur in der eingestellten Messvarianz $\sigma_z$ unterschieden haben. Da die Gossip-Rundenanzahl $T$ nicht direkt von der Präzisionsstufe $f_x$ (Nachkommastellen) der Schätzungen abhängt, konnten wir diese in allen Szenarien gleich halten.

2 Wie auch bei der Evaluation von $\pi_{\mathrm{IF}}$ wurde der Schlüssel zwecks kürzeren Laufzeiten der Simulation hart im Code verdrahtet. Dies, ebenso wie seine Länge, entspricht in keinem Fall den Sicherheitsanforderungen an eine reale Applikation.

| Präz.-Stufe | RMSE | | |
|---|---|---|---|
| | $\sigma_z = 2.5\,°\mathrm{C}$ | $\sigma_z = 5\,°\mathrm{C}$ | $\sigma_z = 10\,°\mathrm{C}$ |
| float | 0.326 621 853 | 0.650 652 659 | 1.303 039 807 |
| 8 Bit | 0.325 446 538 | 0.648 288 379 | 1.298 534 501 |
| 16 Bit | 0.325 447 326 | 0.648 288 277 | 1.298 534 788 |
| 24 Bit | 0.325 447 324 | 0.648 288 276 | 1.298 534 787 |

**Tabelle 5.1:** Ergebnisse der Evaluation des Protokolls $\pi_{\mathrm{GCR}}$ mit verschiedenen Messvarianzen $\sigma_z$.

Die Simulation wurde je Szenario 1000 Mal ausgeführt, wobei jeder Lauf $K = 50$ Schätzungen produzierte und $K \cdot T = 1000$ Gossip-Runden beinhaltete. Für die Schätzungen wurde dann der RMSE bzgl. des tatsächlichen simulierten Systemzustands berechnet, der in der Tabelle 5.1 dargestellt ist.

Wir erkennen, dass die Ausführung des Konsensfilters in jedem beschriebenen Szenario einen etwa 8-fachen Genauigkeitsvorteil gegenüber rohen Messungen bringt. Die konkrete Quantisierungsstufe der Schätzungen hat offensichtlich einen minimalen Einfluss auf ihre Genauigkeit, aber gegenüber einer unverschlüsselten Filterung bringt die Quantisierung einen geringen, aber erkennbaren Vorteil. Das letztere ist vermutlich darauf zurückzuführen, dass in der Gleitkommaarithmetik die Zahlen nach jeder Operation gerundet werden, während in der homomorphen Verarbeitung alle Nachkommastellen in den Chiffraten bis zur Entschlüsselung und der Rückquantisierung erhalten bleiben.

## 5.6 Bewertung

**Sicherheit** Im Abschnitt 5.1 haben wir zwei ideale Funktionalitäten für ein verteiltes Gossip-Konsensfilter aufgestellt, eins mit reellen Eingaben und eins mit Eingaben in endlichen Körpern, und in den Abschnitten 5.2 und 5.3 unter bestimmten Annahmen bewiesen, dass die Protokolle 5.1 ($\pi_{\mathrm{GCR}}$) und 5.2 ($\pi_{\mathrm{GCZ}}$) die entsprechende Funktionalitäten in der Anwesenheit der statischen semiehrlichen PPT-Angreifern sicher berechnen.

Die Sicherheitsimplikationen des verwendeten Angreifermodells und der Annahmen über das Verschlüsselungsschema wurden bereits ausführlich im Abschnitt 4.7 diskutiert und gelten im vollen Maße auch für $\pi_{\mathrm{GCR}}$ bzw. $\pi_{\mathrm{GCZ}}$. Die Korruptionsstrategieannahme in beiden Beweisen 5.2 und 5.4 ist der Annahme b) in dem Satz 4.2 analog: der Angreifer darf nicht eine schlüsselerzeugende und eine homomorph aggregierende Partei gleichzeitig korrumpieren können, sonst lernt er trivial die Eingaben der Nachbarn der letzteren. Die Konsequenzen einer solchen Korruption sind aber im

$\pi_{\mathrm{GCR}}$ bzw. $\pi_{\mathrm{GCZ}}$ weniger gravierend als in $\pi_{\mathrm{IF}}$, weil der Knotengrad der Sensoren in den ersteren deutlich kleiner als der von den Hubs im letzteren ist.

Der im Abschnitt 4.7 beschriebene Dateninjektionsangriff gegen $\pi_{\mathrm{IF}}$ würde auch gegen $\pi_{\mathrm{GCR}}$ funktionieren. Dank der iterativen Berechnung des Konsensmittelwerts würde es aber zur gezielten Verfälschung des Endergebnisses nicht mehr ausreichen, nur die initiale Eingabe eines korrumpierten Sensor zu manipulieren, weil diese Eingabe jede Gossip-Runde immer schwächer in den Konsenswert einfließt. Der korrumpierte Sensor müsste stattdessen in jeder Gossip-Runde seine manipulierte Eingabe immer wieder an die Nachbarn senden, statt ihre Schätzungen protokollkonform zu fusionieren. Dafür muss der Angreifer mindestens boshaft (aktiv) sein, denn ein verstärkter semiehrlicher könnte nur die initiale Eingabe manipulieren. Einen derart korrumpierten Sensor könnte der Kontrolleur möglicherweise ausfindig machen, indem er von mehreren Knoten (statt von nur einem) die Schätzung abfragt und sie auf große Diskrepanzen prüft.

Weil die Gossip-Rundenanzahl im $\pi_{\mathrm{GCR}}$ so gewählt ist, dass dabei die volle Klartextlänge des Chiffrats ausgenutzt wird, könnte aber jeder verstärkte semiehrliche oder boshafte Angreifer das Endergebnis davon unbrauchbar machen. Dazu injiziert der Angreifer eine Eingabe, deren Länge viel größer als $l_x$ ist, und erzeugt dadurch einen modularen Überlauf im Chiffrat $c^i_{T+1}$, der erst nach dem Entschlüsseln auffällt. Im $\pi_{\mathrm{GCZ}}$ dagegen wäre dieser Angriff unmöglich, weil es die modularen Überläufe explizit beachtet und sogar ausnutzt. Der analoge Angriff auf $\pi_{\mathrm{GCZ}}$ würde stattdessen die modulare Gewichte verändern, um die Konvergenz zum Konsenswert zu verhindern.

**Effizienz** Im Abschnitt 5.4 haben wir gezeigt, dass der zusätzliche Rechenaufwand, den die Sensorknoten und der Kontrolleur im $\pi_{\mathrm{GCR}}$ bzw. $\pi_{\mathrm{GCZ}}$ zum Verschlüsseln, Entschlüsseln und Aggregieren der Daten erbringen, linear in der Zustandsvektorlänge $L$, Gossip-Rundenanzal $T$, und der durchschnittlichen Nachbarzahl (Knotengrad) $\hat{\mathcal{N}}$ und konstant in der Netzwerkgröße $N$ ist. Die größte Hürde für die praktische Umsetzung des $\pi_{\mathrm{GCZ}}$ ist aber der im Allgemeinen exponentielle (in $N$) Rechenaufwand zur Berechnung von $FindModWeights\,(\mathcal{G}, s)$, wofür in der Literatur [65, 30] noch keine Lösung gefunden wurde.

Der Netzwerkaufwand in der Setup-Phase ist $O(N)$ im $\pi_{\mathrm{GCR}}$ (Schlüsselverteilung) bzw. $O(N{\cdot}\hat{\mathcal{N}})$ im $\pi_{\mathrm{GCZ}}$ (Verteilung der Gewichtungsvektoren). Der Gesamtnetzwerkaufwand jede Gossip-Runde ist in beiden Protokollen $O(NL\hat{\mathcal{N}})$. Daraus schließen wir, dass das Protokoll $\pi_{\mathrm{GCR}}$ in der Netzwerkgröße $N$ besser als $\pi_{\mathrm{GCZ}}$ skaliert, und auch wenn $\pi_{\mathrm{GCZ}}$ eine stärkere Konvergenzgarantie gibt, es nach dem aktuellen Stand der Forschung vom Rechenaufwand her nicht praktikabel ist.

**Genauigkeit** Im Abschnitt 5.5 haben wir das Protokoll $\pi_{\mathrm{GCR}}$ numerisch simuliert und die Genauigkeit seiner Ergebnisse evaluiert. Im Gegensatz zu $\pi_{\mathrm{IF}}$ bzw. $\pi_{\mathrm{nIF}}$ haben wir dabei beobachtet, dass die Quantisierung der Eingaben bzgl. einer unverschlüsselten Berechnung eine leichte Verbesserung, statt einer Verschlechterung der Genauigkeit mit sich gebracht hatte. Wir vermuten, dass es daran lag, dass bei den homomorphen Evaluationen die komplette Klartextlänge ausgenutzt wurde, um alle Nachkommastellen im Chiffrat zu erhalten, und der Rundungsfehler dadurch nur im ersten (Quantisierung) und im letzten Protokollschritt (Entschlüsselung und Normalisierung) entstand, statt in jeder Gossip-Runde wie bei der Rechnung im Klartext. Die konkrete Präzisionsstufe bei der Quantisierung der Eingaben (Schätzungen und Gewichte) hatte dagegen auf die Genauigkeit nur eine minimale Auswirkung.

KAPITEL 6

# Ergebnisse

**Fazit** In der vorliegenden Arbeit haben wir die Fragestellungen der vertraulichen Signalverarbeitung im Kontext des Kalman-Filters sowie seiner Varianten, nämlich Informations- und Konsensfilter, untersucht. Im Kapitel 2 wurde eine Übersicht der grundlegenden Filterverfahren und der verwandten Arbeiten in dem Bereich der sicheren Signalverarbeitung erstellt. Anschließend haben wir die notwendigen theoretischen Grundlagen aus der formalen Kryptographie zusammengefasst, die für die sichere Signalverarbeitung mithilfe der homomorphen Verschlüsselung relevant sind, und einen Abriss über die aktuelle Forschung auf dem Gebiet gegeben.

Im Kapitel 3 wurde der bisher einzige Vorschlag zur vertraulichen Umsetzung des Kalman-Filters [38] aus der kryptographischen Sicht analysiert, und es wurde gezeigt, wieso er tatsächlich nicht sicher ist. Anhand dieses und anderer Beispiele haben wir die Gefahren erörtert, die mit der *ad hoc* Komposition „sicherer“ kryptographischer Bausteine verbunden sind, wenn davor keine formale Sicherheitsdefinition für das gesamte Protokoll aufgestellt wird. Diese Erkenntnisse motivierten unser Vorgehen beim Entwurf unserer eigenen Protokolle.

Unser erstes neues Protokoll war das vertrauliche verteilte Informationsfilter im Kapitel 4. Wir haben mit einem realistischen Szenario und einem dafür geeigneten Sicherheitsmodell angefangen und einen ausführlichen formalen Beweis der Sicherheit unseres Protokolls ggü. semiehrlichen Angreifern unter gut vertretbaren Annahmen gefunden. Wir haben unser Modell anschließend erweitert, um einem möglichen Sensitivitätsangriff vorzubeugen, und an diesem Beispiel eine kryptographisch sichere Komposition zweier Subprotokolle durchgeführt. Zusätzlich haben wir mit einer Komplexitätsanalyse gute Skalierbarkeit unseres Protokolls gezeigt und eine prototypische Implementierung in Python angefertigt. Damit konnten wir dann eine numerische Evaluation des Verfahrens durchführen und stellten fest, dass es bereits bei minimaler Eingabenquantisierung eine Schätzergenauigkeit aufweist, die mit einer unverschlüsselten Berechnung vergleichbar ist. Unsere vorläufigen Ergebnisse wurden auf der Fusion-Konferenz 2018 in Cambridge vorgestellt [4].

Weil die zentralen Aggregationsstellen ein großer Engpass unseres vertraulichen Informationsfilterprotokolls waren, haben wir im Kapitel 5 zwei Protokolle entworfen, um ein Gossip-Konsensfilter für Sensornetzwerke ohne einen zentralen Knoten umzusetzen. Eins davon basierte auf der Fixpunktarithmetik, das andere nutzte die Synergie zwischen unserem homomorphen Verschlüsselungsschema und dem Konsensfilter in endlichen Körpern [65] aus. Für beide haben wir die Sicherheitsmodelle und -beweise aufgestellt und mittels Komplexitätsanalyse und numerischer Evaluation die Machbarkeit des ersten Ansatzes gezeigt. Für das zweite Protokoll konnten wir keine solche Aussage treffen, weil die effiziente Berechnung der Gewichtungsmatrix für allgemeine Netzwerke aus der Literatur nicht hervorging.

**Ausblick** Im Verlauf unserer Arbeit haben wir immer wieder Fragestellungen entdeckt, deren Bearbeitung ihren Rahmen sprengen würde, die aber ein großes Erkenntnispotenzial für die zukünftige Recherche haben:

- Aus Gründen der Recheneffizienz haben wir bewusst keine vollhomomorphen Verschlüsselungsschemas (FHE) betrachtet und dadurch in Kauf genommen, dass uns nur additiver Homomorphismus zur Verfügung steht. Eine große Fragestellung ist aber, inwiefern der Vorteil der algebraischen Flexibilität, die uns die FHE bietet, den höheren Rechenaufwand rechtfertigt. Aus unserer Sicht könnte eine komplette Studie sich nur mit der Evaluation der FHE-Schemas in Bezug auf ihre Performance in den Signalverarbeitungsaufgaben befassen.
- Wir haben uns außerdem frühzeitig auf asymmetrische homomorphe Schemas festgelegt, obwohl symmetrische Schemas theoretisch recheneffizienter sind. Welche Vor- und Nachteile hätte die Verwendung der symmetrischen homomorphen Verschlüsselung in der Signalverarbeitung?
- Die Sicherheit unserer Protokolle wurde nur ggü. semiehrlichen Angreifern bewiesen. Wie können sie gegen stärkere Angreifer abgesichert werden bzw. ist dies für das gewählte Vorgehen überhaupt möglich? V. a. ist die Sicherheit gegen *false data injections* gefragt, die aktive, heimliche und sogar verstärkte semiehrliche Angreifer bisher trivial durchführen könnten.
- Im Kapitel 3 haben wir gesehen, dass die Skalardivision und die Matrixinversion in der verschlüsselten Domäne besonders aufwendig sind. Sind diese mit der homomorphen Verschlüsselung überhaupt effizient umsetzbar?
- Benötigen wir für die sichere Signalverarbeitung eine schwächere Definition der Protokollkorrektheit, die z. B. auf die korrekte Entschlüsselung der niedrigstwertigen Bits einer verschlüsselten Zahl weniger Wert legt? Welche Konsequenzen hätte es für unsere Sicherheitsgarantien?

- Wenn die homomorphe Verschlüsselung sich als unsicher gegen stärkere Angreifer oder funktionell nicht ausreichend (z. B. für die Matrixinversion) erweist, wie können wir das Yao-Protokoll [77] sinnvoll für die Signalverarbeitung verwenden? Ist die Gleitkomma-Protokollsuite aus [29] dafür ausreichend effizient?
- Wie können wir die Chiffrat-Verdichtung [69] benutzen, um effizienter zu rechnen und zu kommunizieren? Welche homomorphen Schemas unterstützen sie?
- Ist die Beschleunigung der Matrixmultiplikation in der verschlüsselten Domäne grundsätzlich ohne Verlust der Sicherheitsgarantien möglich?
- Im Kapitel 4 haben wir angenommen, dass die Messungen unterschiedlicher Zeitrunden miteinander unkorreliert sind, aber in der Realität werden Schätzungen für die Steuerung verwendet und fließen indirekt in die Messungen späterer Zeitrunden ein. Welche Auswirkungen hat es auf unsere Sicherheitsgarantien?
- In den Kapiteln 4 und 5 haben wir angedeutet, dass Datenpannen unterschiedlich kritisch sein können, je nachdem mit wie vielen ehrlichen eine korrumpierte Partei kommuniziert. Gibt es in der Kryptographie eine Möglichkeit, Korruptionsfälle formal nach dem Schweregrad einzustufen?
- Im Kapitel 5 könnten wir kein Prototyp für das Gossip-Filter in endlichen Körpern ($\pi_{\mathrm{GCZ}}$) implementieren, weil die effiziente Vorberechnung der modularen Gewichtungsmatrix für allgemeine Netzwerke nicht definiert war. Ist diese Funktion $FindModWeights$ für realistische Szenarien effizient berechenbar?
- Für den Sicherheitsbeweis von $\pi_{\mathrm{GCZ}}$ haben wir außerdem die Annahme benötigt, dass ein öffentliches Schlüssel aus dem Klartextmodulus erzeugt werden kann. Gilt sie im Paillier-Schema oder in irgendeinem anderen Schema?
- Viele Forschungsgebiete der formalen Sicherheit lagen außerhalb des Umfangs dieser Arbeit, die aber auf einige unserer Fragestellungen eine Antwort geben könnten. Könnte z. B. die sog. „differentielle Vertraulichkeit“ (engl. *differential privacy*) [21, 20] einen besseren Schutz gegen den von uns beschriebenen Sensitivitätsangriff auf das (unmodifizierte) Informationsfilterprotokoll sein?

Diese Fragen sind nur eine kleine Auswahl der Forschungsthemen auf dem Gebiet der sicheren Signalverarbeitung, die in der ständig stärker vernetzten und digitalisierten Welt immer mehr an Bedeutung und Dringlichkeit gewinnt. Wir sind der festen Überzeugung, dass sie eine der primären Forschungsrichtungen der nahen Zukunft sein wird und dass die Grundlagen, die wir und viele andere Forscher aktuell legen, das Feld maßgeblich gestalten werden.

# Literaturverzeichnis


[1] A. Acar, H. Aksu, A. S. Uluagac, and M. Conti. A survey on homomorphic encryption schemes: Theory and implementation. *arXiv preprint arXiv:1704.03578*, 2017.

[2] C. Aguilar-Melchor, S. Fau, C. Fontaine, G. Gogniat, and R. Sirdey. Recent advances in homomorphic encryption: A possible future for signal processing in the encrypted domain. *IEEE Signal Processing Magazine*, 30(2):108–117, 2013.

[3] M. Aristov. Datenschutzerhaltende Signalverarbeitung: Effizientes Rechnen mit verschlüsselten Daten. In *Von Big Data zu Data Science – Moderne Methoden der Informationsverarbeitung*. Institut für Anthropomatik und Robotik, Lehrstuhl für Intelligente Sensor-Aktor-Systeme (ISAS), Karlsruhe Institute of Technology, 2017.

[4] M. Aristov, B. Noack, U. D. Hanebeck, and J. Müller-Quade. Encrypted Multisensor Information Filtering. In *Proceedings of the 21st International Conference on Information Fusion (Fusion 2018)*, Cambridge, United Kingdom, July 2018.

[5] E. Barker. Recommendation for key management, part 1: General. *NIST special publication*, 800(57):1–147, 2012.

[6] M. Barni, T. Bianchi, D. Catalano, M. Di Raimondo, R. D. Labati, P. Failla, D. Fiore, R. Lazzeretti, V. Piuri, A. Piva, et al. A privacy-compliant fingerprint recognition system based on homomorphic encryption and fingercode templates. In *Biometrics: theory applications and systems (BTAS), Fourth IEEE International Conference on*, pages 1–7. IEEE, 2010.

[7] M. Barni, P. Failla, R. Lazzeretti, A. Paus, A.-R. Sadeghi, T. Schneider, and V. Kolesnikov. Efficient privacy-preserving classification of ECG signals. In *Information Forensics and Security. First IEEE International Workshop on*, pages 91–95. IEEE, 2009.

[8] M. Barni, P. Failla, R. Lazzeretti, A.-R. Sadeghi, and T. Schneider. Privacy-

preserving ECG classification with branching programs and neural networks. *IEEE Transactions on Information Forensics and Security*, 6(2):452–468, 2011.

[9] M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations among notions of security for public-key encryption schemes. In H. Krawczyk, editor, *Advances in Cryptology — CRYPTO '98*, pages 26–45, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.

[10] T. Bianchi and A. Piva. Secure watermarking for multimedia content protection: A review of its benefits and open issues. *IEEE signal processing magazine*, 30(2):87–96, 2013.

[11] T. Bianchi, A. Piva, and M. Barni. Implementing the discrete Fourier transform in the encrypted domain. In *Acoustics, Speech and Signal Processing. IEEE International Conference on*, pages 1757–1760. IEEE, 2008.

[12] T. Bianchi, A. Piva, and M. Barni. Encrypted domain DCT based on homomorphic cryptosystems. *EURASIP Journal on Information Security*, 2009:1, 2009.

[13] A. Boho, G. Van Wallendael, A. Dooms, J. De Cock, G. Braeckman, P. Schelkens, B. Preneel, and R. Van de Walle. End-to-end security for video distribution: The combination of encryption, watermarking, and video adaptation. *IEEE signal processing magazine*, 30(2):97–107, 2013.

[14] Bundesamt für Sicherheit in der Informationstechnik. Kryptographische Verfahren: Empfehlungen und Schlüssellangen, Jan. 2018.

[15] P. Bunn and R. Ostrovsky. Secure two-party k-means clustering. In *Proceedings of the 14th ACM conference on Computer and communications security*, pages 486–497. ACM, 2007.

[16] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *Foundations of Computer Science, 42nd IEEE Symposium on*, pages 136–145. IEEE, 2001.

[17] P. Comesana, L. Perez-Freire, and F. Perez-Gonzalez. Blind newton sensitivity attack. *IEE Proceedings - Information Security*, 153(3):115–125, Sept. 2006.

[18] R. Cramer and V. Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In H. Krawczyk, editor, *Advances in Cryptology — CRYPTO '98*, pages 13–25, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.

[19] Y. Du, L. Gustafson, D. Huang, and K. Peterson. Implementing ML Algorithms with HE. MIT Course 6.857: Computer and Network Security, 2017.

[20] C. Dwork. Differential privacy: A survey of results. In *International Conference on Theory and Applications of Models of Computation*, pages 1–19. Springer, 2008.

[21] C. Dwork and F. McSherry. Differential data privacy, 2010. US Patent.

[22] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE transactions on information theory*, 31(4):469–472, 1985.

[23] Z. Erkin, M. Franz, J. Guajardo, S. Katzenbeisser, I. Lagendijk, and T. Toft. Privacy-preserving face recognition. In *International Symposium on Privacy Enhancing Technologies Symposium*, pages 235–253. Springer, 2009.

[24] Z. Erkin, J. R. Troncoso-Pastoriza, R. L. Lagendijk, and F. Perez-Gonzalez. Privacy-preserving data aggregation in smart metering systems: An overview. *IEEE Signal Processing Magazine*, 30(2):75–86, 2013.

[25] Z. Erkin, T. Veugen, T. Toft, and R. L. Lagendijk. Privacy-preserving user clustering in a social network. In *Information Forensics and Security, 2009. WIFS 2009. First IEEE International Workshop on*, pages 96–100. IEEE, 2009.

[26] P. M. Esperança, L. J. M. Aslett, and C. C. Holmes. Encrypted accelerated least squares regression. *arXiv:1703.00839 [cs, stat]*, Mar. 2017. arXiv: 1703.00839.

[27] P. Failla and M. Barni. Gram-Schmidt orthogonalization on encrypted vectors. In *Trustworthy Internet*, pages 93–103. Springer, 2011.

[28] Federal Information Processing Standards Publication. Announcing the Advanced Encryption Standard (AES), 2001.

[29] M. Franz and S. Katzenbeisser. Processing encrypted floating point signals. In *Proceedings of the Thirteenth ACM Multimedia Workshop on Multimedia and Security*, MM Sec '11, pages 103–108, New York, USA, 2011. ACM.

[30] N. M. Freris and P. Patrinos. Distributed computing over encrypted data. In *Communication, Control, and Computing (Allerton), 2016 54th Annual Allerton Conference on*, pages 1116–1122. IEEE, 2016.

[31] F. Garin and L. Schenato. A survey on distributed estimation and control applications using linear consensus algorithms. In *Networked control systems*, pages 75–107. Springer, 2010.

[32] C. Gentry. *A Fully Homomorphic Encryption Scheme*. PhD thesis, Stanford University, Sept. 2009.

[33] C. Gentry. Fully homomorphic encryption using ideal lattices. In *STOC'09*. Stanford University and IBM Watson, May 2009.

[34] C. Gentry. Computing arbitrary functions of encrypted data. *Communications of the ACM*, 53(3):97–105, 2010.

[35] O. Goldreich. *Foundations of Cryptography*, volume 2: Basic applications. Cambridge Univ. Press, Cambridge, 1. publ. edition, 2004.

[36] O. Goldreich and H. Krawczyk. On the composition of zero-knowledge proof systems. In *International Colloquium on Automata, Languages, and Programming*, pages 268–282. Springer, 1990.

[37] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.

[38] F. J. Gonzalez-Serrano, A. Amor-Martın, and J. Casamayon-Anton. State estimation using an extended Kalman filter with privacy-protected observed inputs. In *2014 IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 54–59. IEEE, Dec. 2014.

[39] S. Halevi. Homomorphic encryption. In Y. Lindell, editor, *Tutorials on the Foundations of Cryptography: Dedicated to Oded Goldreich*, Information Security and Cryptography, pages 219–276. Springer, 2017.

[40] U. D. Hanebeck and O. Schrempf. *Stochastische Informationsverarbeitung*. Institut für Anthropomatik und Robotik, Lehrstuhl für Intelligente Sensor-Aktor-Systeme (ISAS), Karlsruhe Institute of Technology, 2017.

[41] C. Hazay and Y. Lindell. *Efficient Secure Two-Party Protocols*. Springer Berlin Heidelberg, 2010.

[42] Institute of Electrical and Electronics Engineers. IEEE standard for floating-point arithmetic. *IEEE Std 754-2008*, pages 1–70, Aug 2008.

[43] G. Jagannathan, K. Pillaipakkamnatt, and R. N. Wright. A new privacy-preserving distributed k-clustering algorithm. In *Proceedings of the 2006 SIAM International Conference on Data Mining*, pages 494–498. SIAM, 2006.

[44] G. Jagannathan and R. N. Wright. Privacy-preserving distributed k-means clustering over arbitrarily partitioned data. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 593–599. ACM, 2005.

[45] S. J. Julier and J. K. Uhlmann. Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, 92(3):401–422, 2004.

[46] B. Kaliski and J. Staddon. PKCS# 1: RSA cryptography specifications version 2.0. Technical report, The Internet Society, 1998.

[47] R. E. Kalman. A New Approach to Linear Filtering and Prediction Problems. *Transactions of the ASME - Journal of Basic Engineering*, 82:35–45, 1960.

[48] J. Katz and Y. Lindell. *Introduction to Modern Cryptography*. CRC Press, 2nd edition, 2015.

[49] R. L. Lagendijk, Z. Erkin, and M. Barni. Encrypted signal processing for privacy protection: Conveying the utility of homomorphic encryption and multiparty computation. *IEEE Signal Processing Magazine*, 30(1):82–105, Jan. 2013.

[50] A. K. Lenstra. Key length. Contribution to the Handbook of Information Security, 2004.

[51] H. Lin, J. Shao, C. Zhang, and Y. Fang. CAM: cloud-assisted privacy preserving mobile health monitoring. *IEEE Transactions on Information Forensics and Security*, 8(6):985–997, 2013.

[52] Y. Lindell. How to simulate it – a tutorial on the simulation proof technique. In Y. Lindell, editor, *Tutorials on the Foundations of Cryptography: Dedicated to Oded Goldreich*, Information Security and Cryptography, pages 277–346. Springer, 2017.

[53] Y. Liu, P. Ning, and M. K. Reiter. False data injection attacks against state estimation in electric power grids. *ACM Transactions on Information and System Security (TISSEC)*, 14(1):13, 2011.

[54] Y. Luo, S.-c. S. Cheung, T. Pignata, R. Lazzeretti, and M. Barni. An efficient protocol for private iris-code matching by means of garbled circuits. In *Image Processing (ICIP), 19th IEEE International Conference on*, pages 2653–2656. IEEE, 2012.

[55] R. Marchthaler and S. Dingler. *Kalman-Filter: Einführung in die Zustandsschätzung und ihre Anwendung für eingebettete Systeme*. Springer Vieweg, 2017.

[56] MathWorks. Fixed-point arithmetic. MATLAB & Simulink, 2018.

[57] M. Meyer. *Signalverarbeitung: Analoge und digitale Signale, Systeme und Filter*. Springer-Verlag, 8. updated edition, 2017.

[58] D. Micciancio and M. Walter. On the bit security of cryptographic primitives. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 3–28. Springer, 2018.

[59] Y. Mo, E. Garone, A. Casavola, and B. Sinopoli. False data injection attacks against state estimation in wireless sensor networks. In *Decision and Control (CDC), 2010 49th IEEE Conference on*, pages 5967–5972. IEEE, 2010.

[60] B. Noack, U. D. Hanebeck, G. Kurz, D. Brunn, M. Huber, M. Baum, A. Zea, and F. Rosenthal. *Lokalisierung mobiler Agenten*. Institut für Anthropomatik und Robotik, Lehrstuhl für Intelligente Sensor-Aktor-Systeme (ISAS), Karlsruhe Institute of Technology, 2018.

[61] B. Noack, J. Sijs, M. Reinhardt, and U. D. Hanebeck. Treatment of dependent information in multisensor Kalman filtering and data fusion. In H. Fourati, editor, *Multisensor Data Fusion: From Algorithms and Architectural Design to Applications*, pages 169–192. CRC Press, 2015.

[62] R. Olfati-Saber. Distributed Kalman filter with embedded consensus filters. In *Proceedings of the 44th IEEE Conference on Decision and Control and European Control Conference (CDC-ECC'05)*, pages 8179–8184. IEEE, 2005.

[63] R. Olfati-Saber. Distributed Kalman filtering for sensor networks. In *Proceedings of the 46th IEEE Conference on Decision and Control (CDC'07)*, pages 5492–5498. IEEE, 2007.

[64] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 223–238. Springer, 1999.

[65] F. Pasqualetti, D. Borra, and F. Bullo. Consensus networks over finite fields. *Automatica*, 50(2):349–358, 2014.

[66] S. Rane and P. T. Boufounos. Privacy-preserving nearest neighbor methods: Comparing signals without revealing them. *IEEE Signal Processing Magazine*, 30(2):18–28, 2013.

[67] M. Ruan, M. Ahmad, and Y. Wang. Secure and privacy-preserving average consensus. In *Proceedings of the 2017 Workshop on Cyber-Physical Systems Security and Privacy*, pages 123–129. ACM, 2017.

[68] A.-R. Sadeghi, T. Schneider, and I. Wehrenberg. Efficient privacy-preserving face recognition. In *International Conference on Information Security and Cryptology*, pages 229–244. Springer, 2009.

[69] N. P. Smart and F. Vercauteren. Fully homomorphic SIMD operations. *Designs, Codes and Cryptography*, 71(1):57–81, 2014.

[70] J. R. Troncoso-Pastoriza, S. Katzenbeisser, and M. Celik. Privacy preserving error resilient dna searching through oblivious automata. In *Proceedings of the*

*14th ACM conference on Computer and communications security*, pages 519–528. ACM, 2007.

[71] J. Vaidya and C. Clifton. Privacy-preserving k-means clustering over vertically partitioned data. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 206–215. ACM, 2003.

[72] T. Veugen. Comparing encrypted data. Multimedia Signal Processing Group, Delft University of Technology, and TNO Information and Communication Technology, Delft, The Netherlands, 2011.

[73] D. Wagner. Resilient aggregation in sensor networks. In *Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks*, pages 78–87. ACM, 2004.

[74] R. Wang, X. Wang, Z. Li, H. Tang, M. K. Reiter, and Z. Dong. Privacy-preserving genomic computation through program specialization. In *Proceedings of the 16th ACM conference on Computer and communications security*, pages 338–347. ACM, 2009.

[75] T. Winkler and B. Rinner. Security and privacy protection in visual sensor networks: A survey. *ACM Comput. Surv.*, 47(1):2:1–2:42, May 2014.

[76] L. Xiao, O. Bastani, and I.-L. Yen. An efficient homomorphic encryption protocol for multi-user systems. *IACR Cryptology ePrint Archive*, 2012.

[77] A. Yao. How to generate and exchange secrets. In *Foundations of Computer Science, 27th Annual Symposium on*, pages 162–167. IEEE, 1986.