Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

HOMOMORPHE VERSCHLÜSSELUNG

Verfahren zum Rechnen auf verschlüsselten Daten

8. Oktober 2014

Viola Campos

Fachbereich Design Informatik Medien
Hochschule RheinMain

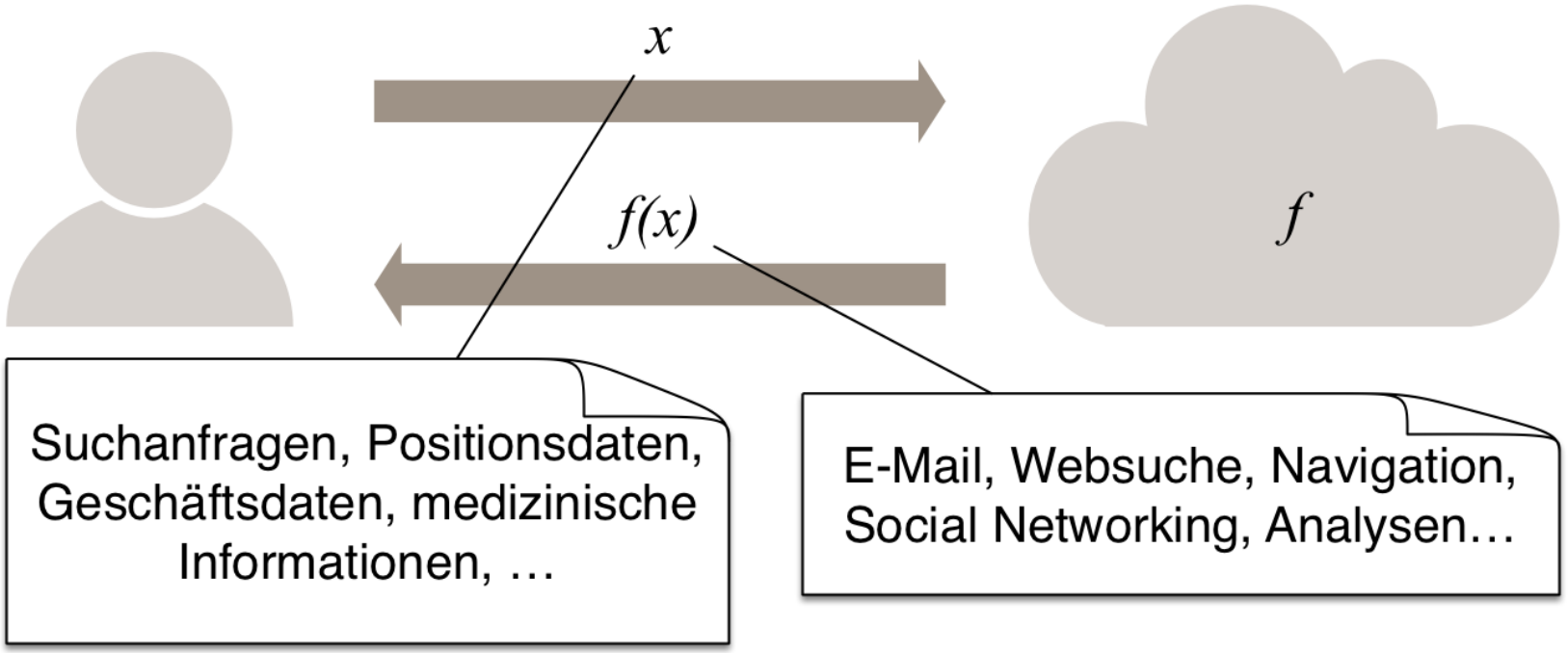
GLIEDERUNG

1. Einleitung
2. Homomorphe Verschlüsselung
3. Das Approximate Eigenvektor Schema
4. Implementierung
5. Ausblick

EINLEITUNG

CLOUD COMPUTING

Ziel: Auslagern von Berechnungen in die Cloud



Problem:

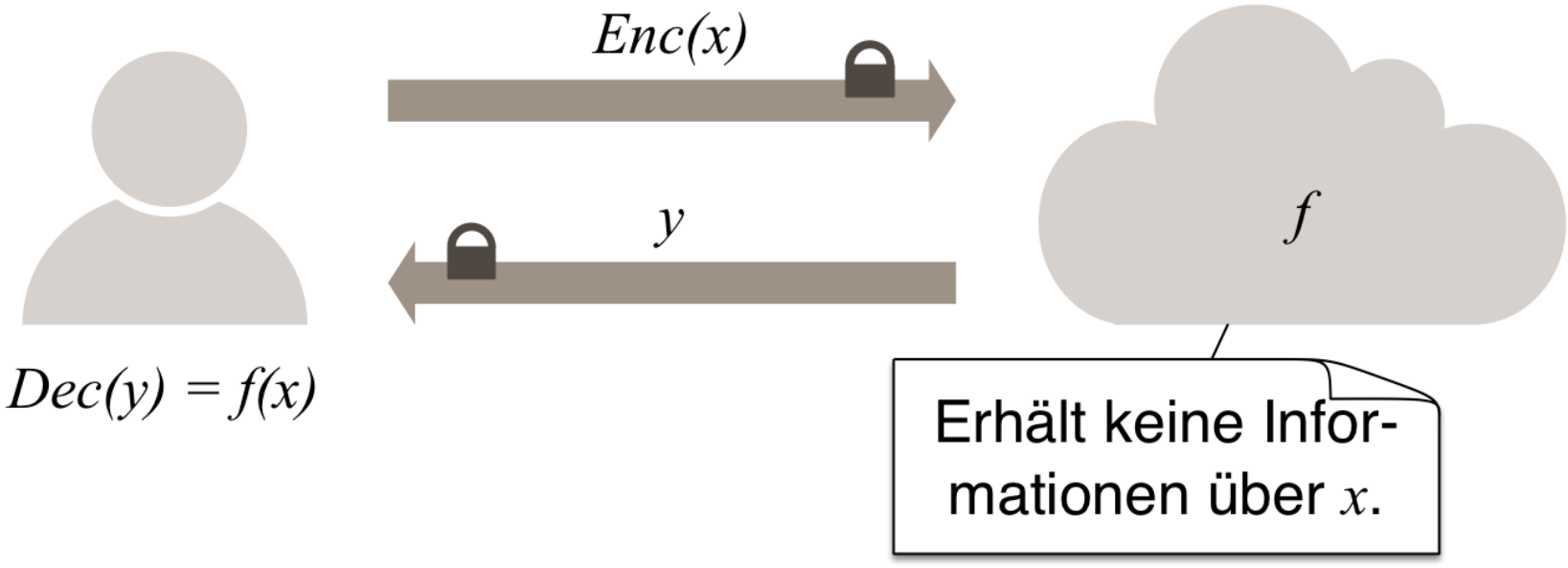
Wie lassen sich vertrauliche Informationen schützen?

SICHERES CLOUD COMPUTING

Idee: Ich möchte die Verarbeitung meiner Daten auslagern ohne den Zugriff auf diese zu erlauben.

Copyright by NASA

SICHERES CLOUD COMPUTING

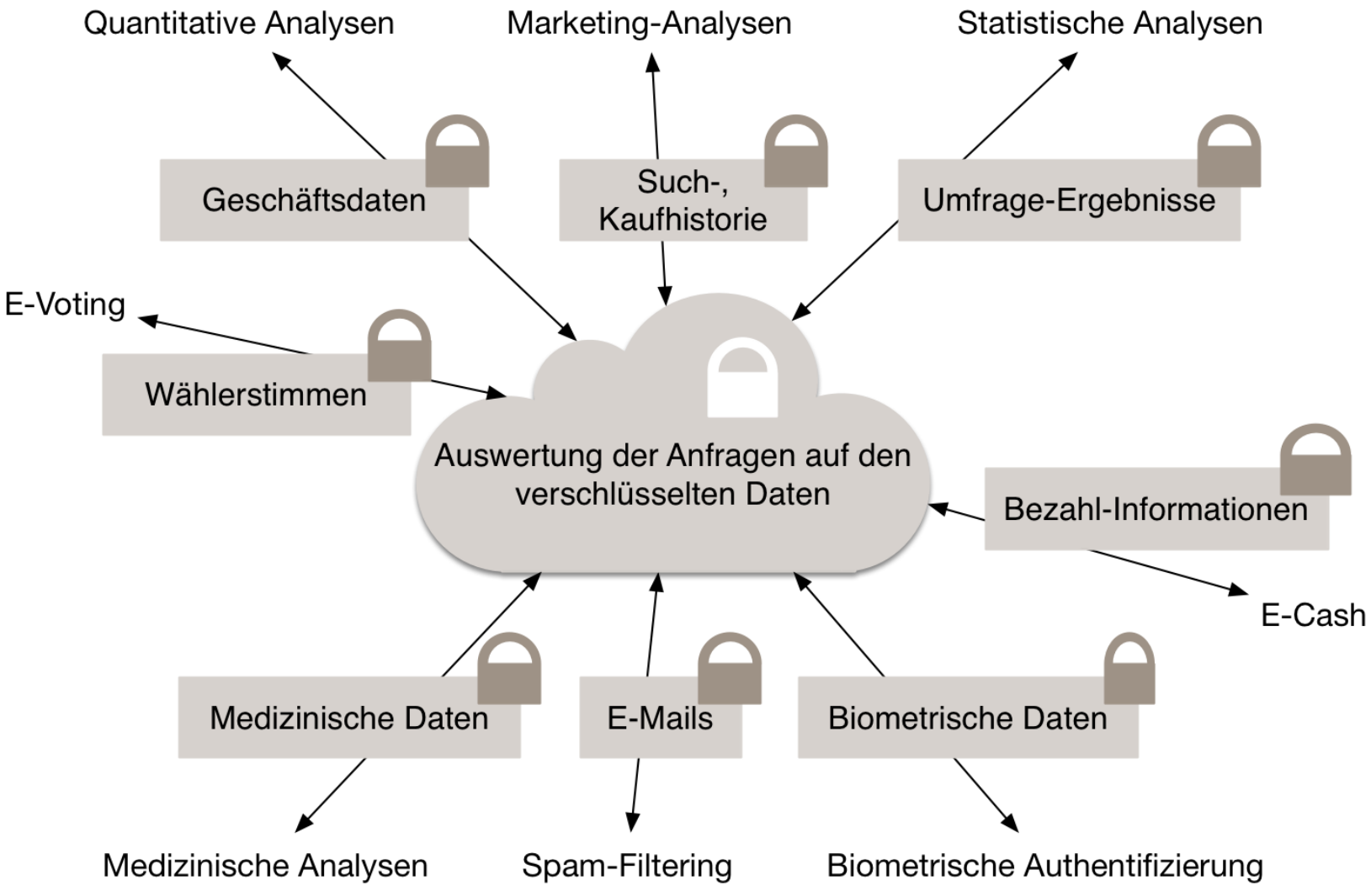


Gesucht:

Möglichkeit, Funktion mit verschlüsselten Argumenten auszuwerten:

$$Eval(f, Enc(x)) \rightarrow Enc(f(x))$$

ANWENDUNGSMÖGLICHKEITEN



ENTWICKLUNG

Entwicklung homomorpher Verschlüsselung

1978	Definition homomorpher Verschlüsselung *Rivest, Adleman, Dertouzos*
bis 2008	Teilweise homomorphe Verschlüsselungssysteme (homomorph bzgl. Addition oder Multiplikation, keine Kombination)
2009	Erstes voll homomorphes Verschlüsselungssystem *Gentry*.
seit 2009	Varianten und Weiterentwicklungen von Gentrys System.

HOMOMORPHE VERSCHLÜSSELUNG

HOMOMORPHIE

Definition

Als **Homomorphismus** oder **Homomorphie** bezeichnet man in der Algebra eine strukturerhaltende Abbildung zwischen zwei algebraischen Strukturen vom selben Typ.

Beispiel: Für zwei Ringe $(R, +, \cdot)$ und $(S, \oplus, \otimes)$ wird die Abbildung $f : R \to S$ Ringhomomorphismus genannt, wenn für jedes Element a, b von R gilt:

$$f(a + b) = f(a) \oplus f(b) \text{ und}$$
$$f(a \cdot b) = f(a) \otimes f(b)$$

Homomorphes Verschlüsselungssystem:

$$\text{Enc}(m_1 + m_2) = \text{Enc}(m_1) \oplus \text{Enc}(m_2)$$
$$\text{Enc}(m_1 \cdot m_2) = \text{Enc}(m_1) \otimes \text{Enc}(m_2)$$

HOMOMORPHE VERSCHLÜSSELUNG 1/2

Definition

Ein **homomorphes Verschlüsselungssystem** ist ein 4-Tupel der folgenden Algorithmen:

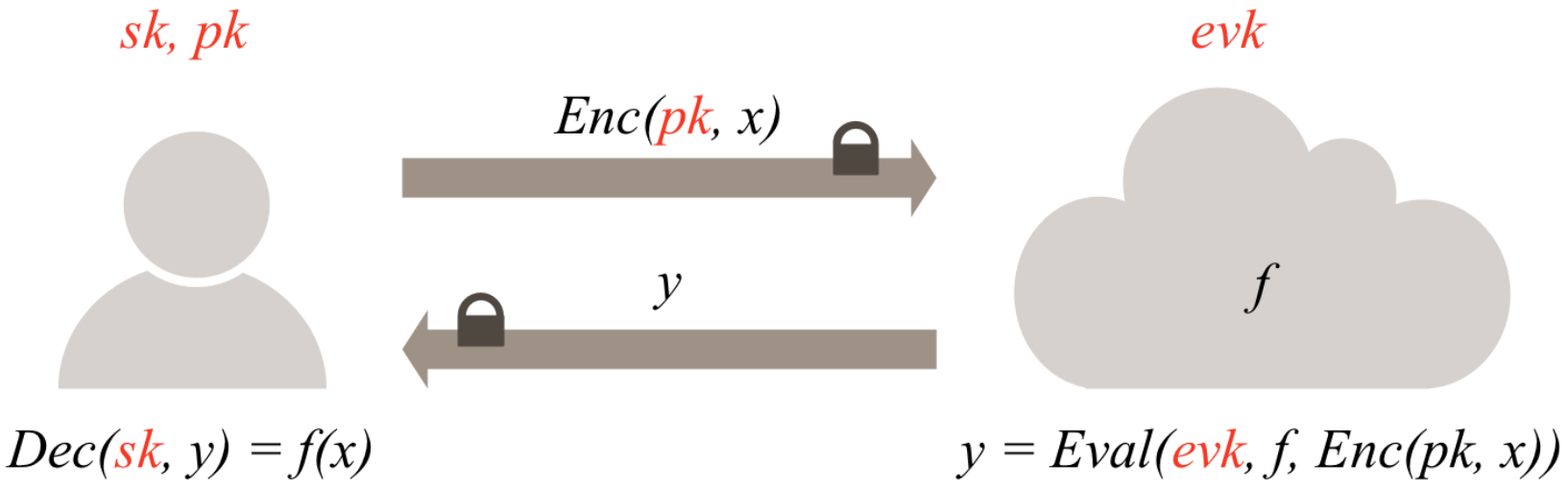
Schlüsselerzeugung:	$\text{Gen}(\lambda^*) \rightarrow (sk, pk, evk^\dagger)$
Verschlüsselung:	$\text{Enc}(pk, m) \rightarrow C$
Entschlüsselung:	$\text{Dec}(sk, C) \rightarrow m$
Homomorphe Berechnung:	$\text{Eval}(evk, f, C_1, C_2, \dots) \rightarrow C'$

---

*Sicherheitsparameter

†Secret Key, Public Key, Evaluation Key

HOMOMORPHE VERSCHLÜSSELUNG 2/2



Anforderungen

Sicherheit, Korrektheit, Kompaktheit

SICHERHEIT

Semantische Sicherheit / IND-CPA‡

Ein Angreifer darf einem Geheimtext C, der entweder die Nachricht m_1 oder m_2 verschlüsselt, den zugrundeliegenden Klartext auch dann nicht zuordnen können, wenn er den öffentlichen Schlüssel kennt und m_1 und m_2 selbst gewählt hat.

Semantisch sicheres Verschlüsselungsverfahren muss **probabilistisch** sein.

Problem: Rauschen wird durch homomorphe Berechnungen verstärkt, ab gewissem Betrag ist korrekte Entschlüsselung unmöglich.

‡ciphertext indistinguishability under chosen plaintext attacks

KORREKTHEIT 1/2

Korrekte Verschlüsselung:

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

Korrekte homomorphe Evaluation:

$$\mathsf{Dec}(sk, \mathsf{Eval}(evk, f, \mathsf{Enc}(pk, m))) = f(m)$$

Korrektheit für vollständiges System logischer Operatoren, z.B. (AND, OR, NOT) oder NAND $\rightarrow$ Korrektheit für beliebige Funktionen

KORREKTHEIT 2/2

- Fully Homomorphic Encryption
  - Korrektheit für **jede** Funktion f
  - Reduzierung des Rauschens nach einigen Operationen
- Leveled Fully Homomorphic Encryption
  - Schlüssel werden so erzeugt, dass System Funktionen **gewünschter Komplexität** korrekt auswerten kann
- Somewhat Homomorphic Encryption
  - Korrekte Auswertung **einfacher** Funktionen f
  - geringere Komplexität als voll homomorphe Systeme, oft ausreichend

DAS APPROXIMATE EIGENVEKTOR SCHEMA

IDEE

Eigenwert und Eigenvektor

Für eine $n \times n$-Matrix $\mathbf{C}$ nennt man einen n-dimensionalen Vektor $\vec{v}$ **Eigenvektor** und einen Skalar m **Eigenwert** von $\mathbf{C}$, wenn gilt:

$$\mathbf{C} \cdot \vec{v} = m \cdot \vec{v}$$

Beobachtung: Für Matrizen $\mathbf{C}_1, \mathbf{C}_2$ mit gemeinsamem Eigenvektor $\vec{v}$ und dazugehörigen Eigenwerten m_1, m_2 gilt:

$$(\mathbf{C}_1 + \mathbf{C}_2) \cdot \vec{v} = (m_1 + m_2) \cdot \vec{v}$$
$$(\mathbf{C}_1 \cdot \mathbf{C}_2) \cdot \vec{v} = (m_1 \cdot m_2) \cdot \vec{v}$$

Idee: $\vec{v} =$ geheimer Schlüssel, $\mathbf{C} =$ Geheimtext, $m =$ Nachricht
$\Rightarrow$ Homomorphie bzgl. Addition und Multiplikation

SICHERHEIT

Problem: System ist unsicher, Eigenvektoren lassen sich leicht bestimmen.
$\Rightarrow$ Geheimtexte müssen einen kleinen zufälligen Fehler $\vec{e}$ enthalten.

$$\mathbf{C} \cdot \vec{v} = m \cdot \vec{v} + \vec{e} \approx m \cdot \vec{v}$$

$\vec{v}$	Geheimer Schlüssel
$\mathbf{C}$	Geheimtext
m	Nachricht
$\vec{e}$	Fehlervektor

Alle Berechnungen finden in $\mathbb{Z}_q$, im Ring der ganzen Zahlen modulo einer positiven ganzen Zahl q statt.

Sicherheit des Systems lässt sich auf ›Learning with Errors‹-Problem zurückführen.

HOMOMORPHE OPERATIONEN

Ziel: Wenn $\mathbf{C}_1$ und $\mathbf{C}_2$ Verschlüsselungen von m_1 und m_2 sind, soll gelten:

$$\mathbf{C}_1 + \mathbf{C}_2 = \mathsf{Enc}(m_1 + m_2)$$

$$\mathbf{C}_1 \cdot \mathbf{C}_2 = \mathsf{Enc}(m_1 \cdot m_2)$$

Zur Erinnerung:

$$\mathbf{C}_1 \cdot \vec{v} = m_1 \vec{v} + \vec{e}_1$$

$$\mathbf{C}_2 \cdot \vec{v} = m_2 \vec{v} + \vec{e}_2$$

$$\begin{aligned}\mathbf{C}^+ \cdot \vec{v} &= (\mathbf{C}_1 + \mathbf{C}_2) \cdot \vec{v} \\ &= \mathbf{C}_1 \vec{v} + \mathbf{C}_2 \vec{v} \\ &= m_1 \vec{v} + \vec{e}_1 + m_2 \vec{v} + \vec{e}_2 \\ &= (m_1 + m_2)\vec{v} + \underbrace{(\vec{e}_1 + \vec{e}_2)}_{\vec{e}_{add}}\end{aligned}$$

$$\begin{aligned}\mathbf{C}^\times \cdot \vec{v} &= \mathbf{C}_1 \cdot \mathbf{C}_2 \cdot \vec{v} \\ &= \mathbf{C}_1 \cdot (m_2 \vec{v} + \vec{e}_2) \\ &= m_2 \cdot (m_1 \vec{v} + \vec{e}_1) + \mathbf{C}_1 \vec{e}_2 \\ &= (m_1 \cdot m_2)\vec{v} + \underbrace{m_2 \vec{e}_1 + \mathbf{C}_1 \vec{e}_2}_{\vec{e}_{mult}}\end{aligned}$$

BEGRENZUNG DES FEHLERS

Möglichkeiten, das Fehlerwachstum während der homomorphen Evaluation zu reduzieren:

- Bitweise Verschlüsselung
  - Nachrichten m stammen aus dem Nachrichtenraum $\{0, 1\}$
- Ciphertext Flattening
  - Zeitweise Zerlegung der Vektor- und Matrix-Elemente in deren binäre Repräsentation
  - reduziert Maximumsnorm $\Rightarrow$ geringeres Fehlerwachstum
- Sequentielle Funktionsauswertung
  - Fehler wächst asymmetrisch während homomorpher Multiplikation
  - geschickter Aufbau der auszuwertenden Schaltkreise
- Bootstrapping

IMPLEMENTIERUNG

ÜBERBLICK

Verwendete Technologien: Python und Computeralgebrasystem Sage

Implementierung in zwei Varianten:

- Approximate Eigenvector System von Gentry, Sahai und Waters
- System mit Optimierungen von Brakerski und Vaikuntanathan

SPEICHERBEDARF

n	q	Private Key	Public Key	Ciphertext
16	16^4	0.56kB	12.75kB	163.12kB
32	32^4	1.69kB	72.18kB	1.14MB
64	64^4	4.76kB	414.37kB	7.55MB
128	128^4	12.78kB	2.20MB	46.71MB
256	256^4	33.12kB	11.54MB	274.38MB
512	512^4	83.41kB	56.69MB	1.51GB

Tabelle: Speicherbedarf eines GSW Systems mit Gitterdimension n und Modulus q. Ciphertext bezeichnet den Geheimtext, der erzeugt wird, um ein Klartext-Bit zu verschlüsseln.

LAUFZEITMESSUNG

Testumgebung: 2.4GHz Core, 8GB RAM, Implementierung in Python, Zeiten in s

n	Level	KeyGen	Enc	Dec	Add	Mult
Implementierung nach GSW13						
16	1	0.03	3.00	0.00	1.53	1.51
32	1	0.10	32.19	0.01	16.10	16.22
64	1	0.62	350.70	0.04	133.14	133.91
128	1	4.86	-	-	-	-
Implementierung mit Optimierungen aus BV14						
16	10	0.02	4.63	0.00	3.75	8.22
32	10	0.07	61.38	0.00	56.88	123.36
64	10	0.59	865.84	0.03	796.75	2878.05
128	10	4.01	-	-	-	-

AUSBLICK

FAZIT

- Implementiertes Schema erreicht leveled homomorphes System durch Wahl passender Systemparameter q[§], n[¶] und e[**].
  - Problem: riesige Parameter und geringere Sicherheit
  - widersprüchliche Anforderungen für Sicherheit und homomorphe Fähigkeiten des Systems
- Nutzung eines sequentiellen Modells (Branching Tree) für die homomorphe Funktionsauswertung reduzierte Fehlerwachstum deutlich
- Schwäche des Systems: Geheimtextexpansion

[§] Modulus von $\mathbb{Z}_q$

[¶] Dimension der Geheimtextmatrizen und Schlüssel

[**] Maximaler Fehlerbetrag

OFFENE PROBLEME

- **Effizientere Systeme**
  - Riesiger Overhead $\frac{Time(Eval(f))}{Time(f)}$ aktueller FHE Systeme
  - hoher Speicherbedarf für Schlüssel und Geheimtexte
- **FHE Systeme basierend auf neuen kryptographischen Annahmen**
  - Alle aktuellen Systeme stammen aus dem Gebiet der gitterbasierten Kryptographie
- **Bounded Malleability**
  - Möglichkeit, nur bestimmte homomorphe Operationen zuzulassen
  - Schutz vor unerlaubten Veränderungen an verschlüsselten Daten
- **Alternative zu Bootstrapping**
  - effizientere Methode der Fehlerreduzierung

VIELEN DANK, NOCH FRAGEN?

BOOTSTRAPPING

Problem:

Fehler in Geheimtexten wächst mit jeder homomorphen Operation. Wie lassen sich beliebig viele Operationen ermöglichen?

Idee: Homomorphe Auswertung der eigenen Entschlüsselungsfunktion reduziert Rauschen ohne Klartext offenzulegen.

Für 2 Schlüsselpaare (sk_1, pk_1) und (sk_2, pk_2) und einen mit pk_1 verschlüsselten Geheimtext C gilt:

$$\mathsf{Eval}(evk, f_{\mathsf{Dec}}, C, \mathsf{Enc}(pk_2, sk_1)) = \mathsf{Enc}(pk_2, m)$$

Fehler im Ergebnis entspricht frisch verschlüsseltem Geheimtext.

ENTSCHLÜSSELUNG

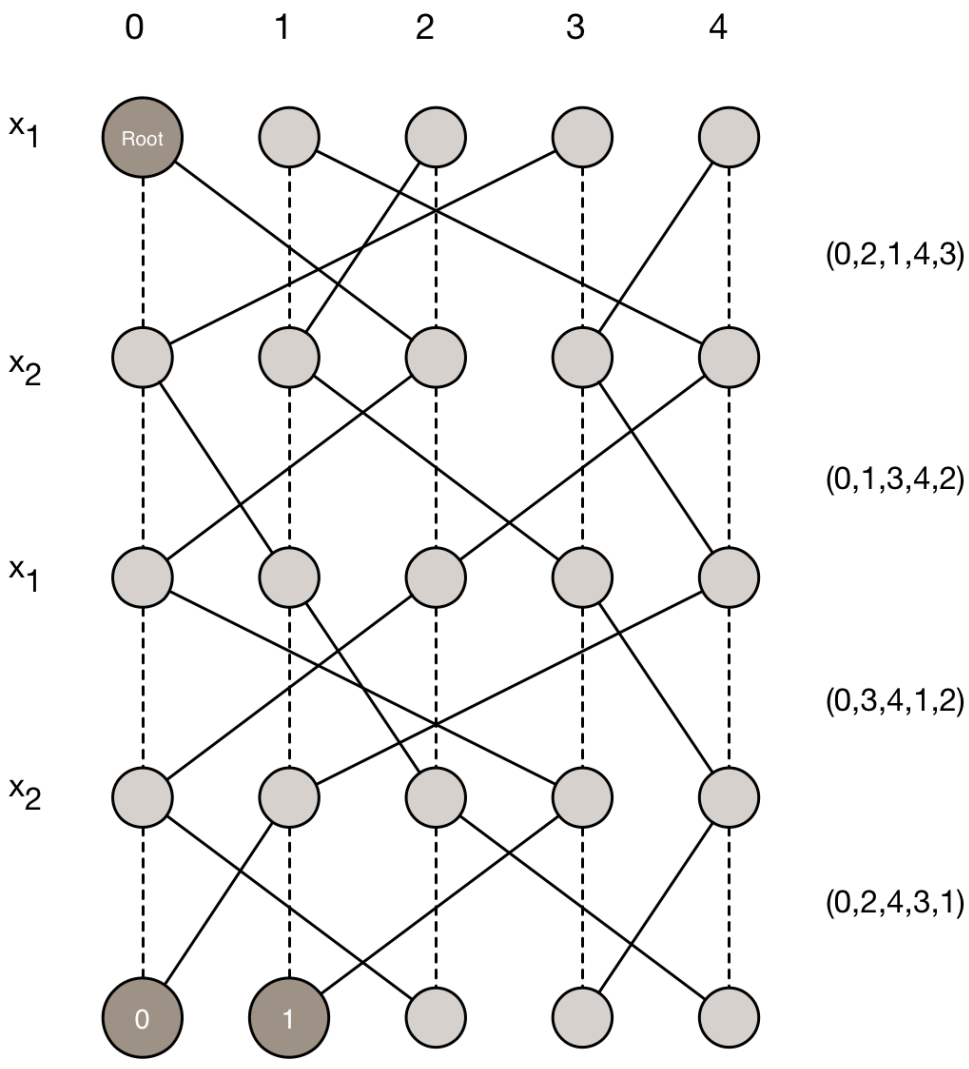
$\mathbf{C}$ verschlüsselt m, wenn $\mathbf{C} \cdot \vec{v} = m \cdot \vec{v} + \vec{e}$

- Alle Berechnungen finden in $\mathbb{Z}_q$, im Ring der ganzen Zahlen modulo einer positiven ganzen Zahl q statt.
- Klartexte werden bitweise verschlüsselt, Nachrichten m stammen also aus $\{0, 1\}$.
- Der geheime Schlüssel $\vec{v}$ enthält ein großes Element $\vec{v}[i] = \frac{q}{2}$.

Dann gilt $(\mathbf{C} \cdot \vec{v})[i] = \frac{q}{2} \cdot m + \vec{e}[i]$ und m lässt sich durch Runden bestimmen.

Bedingung für korrekte Entschlüsselung: $\|\vec{e}\| < \frac{q}{4}$

BRANCHING PROGRAM

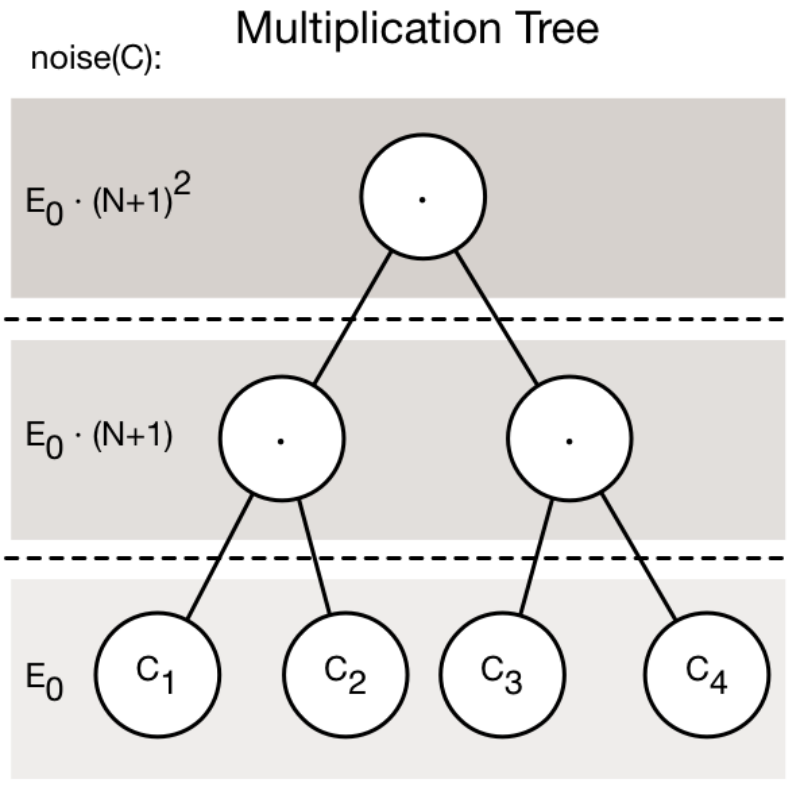


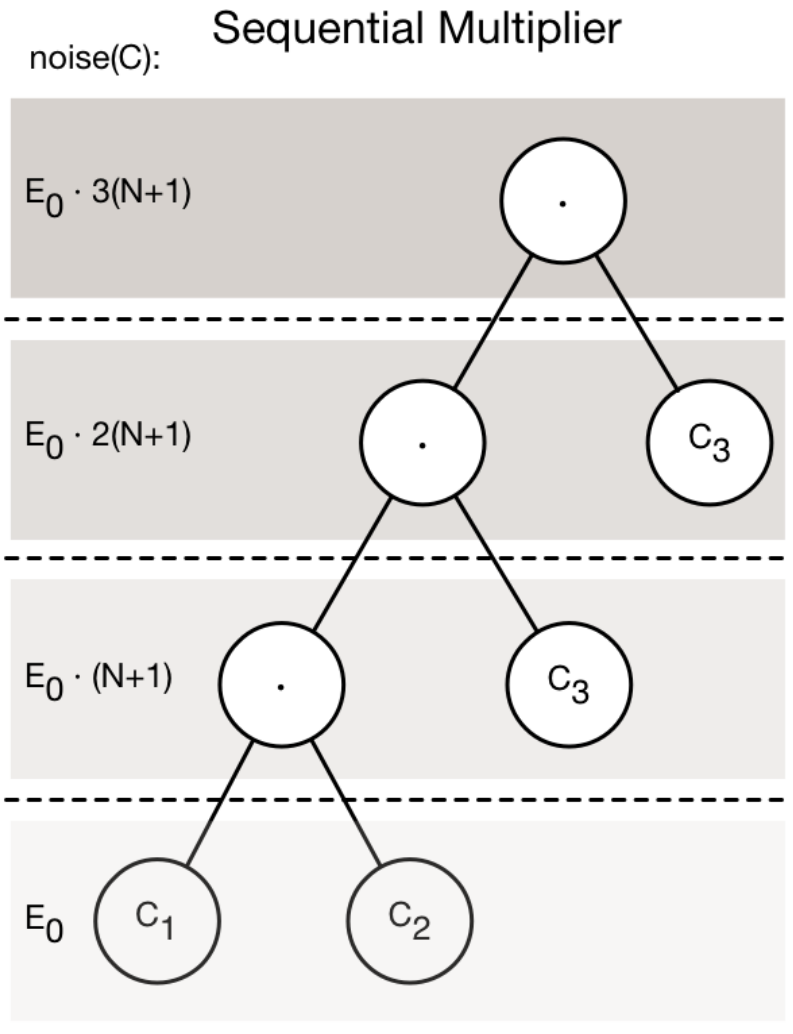
Permutation Branching Program der Breite 5 [?] für eine Funktion $f\colon \{0,1\}^2 \to \{0,1\}$.

Kanten für $x_i = 1$ sind durchgezogen, für $x_i = 0$ gestrichelt dargestellt.

Das Beispiel berechnet $f = \mathrm{AND}(x_1, x_2)$.

SEQUENTIELLE AUSWERTUNG





KOMPAKTHEIT

Ziel: Die Laufzeit von Dec(sk, C) ist für Ausgaben von Enc(pk, m) und Eval(evk, f, C) gleich.

Kompaktheit

- Die Länge der Verschlüsselung eines Bits hängt nur vom Sicherheitsparameter λ ab.
- Insbesondere ist die Länge der Ausgabe von Eval(evk, f, C_1, C_2, ...) unabhängig von der ausgewerteten Funktion f und der Anzahl ihrer Eingabewerte.

OPTIMIERUNGSMÖGLICHKEITEN

- Batching
- Hybride voll homomorphe Verschlüsselung