# Homomorphe Verschlüsselung

Sophie Friedrich, Nicholas Höllermeier, Martin Schwaighofer

11. Juni 2012

# Inhaltsverzeichnis

# Was ist homomorphe Verschlüsselung?

Homomorphe Verschlüsselung ist eine Form der Verschlüsselung die es ermöglicht:

- Berechnungen auf Ciphertexten auszuführen

wobei das entschlüsselte Resultat das gleiche ist, wie wenn man:

- die Cyphertexte entschlüsselt
- auf den entschlüsselten Cyphertexten die Berechnungen durchführt
- das Ergebnis wieder verschlüsselt

# Einfaches Beispiel

Alice eröffnet einen Würstelstand in Salzburg, und beauftragt Mallory wöchentlich ihre Verkaufszahlen zu berechnen. Sie verwendet ein homomorphes Verschlüsselungsverfahren um ihre Umsatzdaten vor Mallory geheim zu halten.
$E(m) = m * k$ und $D(c) = \frac{c}{k}$, wobei $k = 2$ der von Alice gewählte Schlüssel ist.
Alice verkauft von Montag bis Freitag täglich genau ein Würstel, nämlich an Bob. Nur am Mittwoch verkauft sie ihm zwei Würstel.
Sie schickt ihre verschlüsselten Umsätze täglich an Mallory.
Mallory berechnet $2 + 2 + 4 + 2 + 2 = 12$ und sendet diese Zahl am Freitag an Alice zurück.
Alice entschlüsselt mit $D(12) = 12/2 = 6$ und erhält somit das korrekte Ergebnis, ohne Mallory ihre Umsätze zu verraten.

# Motivation

Wozu homomorphe Verschlüsselung?

- Cloud-Computing
  - rechenintensive Anwendungen werden auf Server im Internet ausgelagert
  - große Datenmengen werden extern gespeichert
  - Datensicherheit nicht immer optimal gewährleistet
- E-voting
  - Einhaltung des Wahlgeheimnisses
  - Stimmen müssen aufsummiert und ausgezählt werden,ohne die individuellen Stimmen zu kennen

$\Rightarrow$ Homomorphe Verschlüsselung schafft Abhilfe für diese Probleme

# Gruppe

Eine Gruppe ist ein Paar $(G, \circ)$. G ist eine Menge,$\circ$ eine zweistellige Verknüpfung: $\circ : G \times G \to G, (a, b) \mapsto a \circ b$
Es muss gelten:

- Assoziativität: $a, b, c \in G : (a \circ b) \circ c = a \circ (b \circ c)$
- neutrales Element: $\exists e \in G, \forall a \in G : a \circ e = e \circ a = a$
- inverses Element: $\forall a \in G \ \exists a^{-1} \in G : a \circ a^{-1} = a^{-1} \circ a = e$

Eine Gruppe heißt abelsche Gruppe, wenn gilt: $a \circ b = b \circ a$
$\forall a, b \in G$
d.h. das Kommutativgesetz muss gelten

# Ring

Ein Ring $(R, +, *)$ ist eine Menge R mit zwei inneren binären Verknüpfungen $+$ und $*$. Dabei muss gelten:

- $(R, +)$ ist eine abelsche Gruppe
- Assoziativ bezüglich der Multiplikation:
  $\forall a, b, c \in R : (a * b) * c = a * (b * c)$
- Abgeschlossen bezüglich der Multiplikation:
  $\forall a, b \in R : a * b \in R$
- Distributivgesetze:
  - $a * (b + c) = a * b + a * c$
  - $(a + b) * c = a * c + b * c$

# Gruppenhomomorphismus

Seien $(G, \circ_g)$ und $(F, \circ_f)$ Gruppen, dann heißt die Abbildung $f : G \to F$ Gruppenhomomorphismus, wenn $\forall a, b \in G$ gilt:
$f(a \circ_g b) = f(a) \circ_f f(b)$

Beispiel:
Die Exponentialfunktion zwischen den Gruppen $(\mathbb{R}, +)$ und $(\mathbb{R}^*, *)$ ist ein Gruppenhomomorphismus, da gilt:

$exp(x + y) = exp(x) * exp(y)$

# Homomorphe Verschlüsselung

Ein Verschlüsselungsverfahren mit $E : G \to G'$ (G,G' Gruppen) heißt homomorph, wenn gilt: $\forall m_1, m_2 \in G$ :
$E(m_1 \circ_g m_2) = E(m_1) \circ_{g'} E(m_2)$

Das Verschlüsselungsverfahren heißt additiv homomorph wenn:
$G := (G, +)$ (additive Gruppe)

und multiplikativ homomorph wenn: $G := (G, *)$ (multiplikative Gruppe)

# RSA

RSA ist homomorph bezüglich der Multiplikation:

$E(m_1) = m_1^e \bmod n$ :

$E(m_1) * E(m_2) = m_1^e \bmod n * m_2^e \bmod n = m_1^e * m_2^e \bmod n$
$= (m_1 * m_2)^e \bmod n = E(m_1 * m_2)$

# RSA-Beispiel

- $p = 11$ $q = 13$
- $n = p * q = 143$
- $\phi(n) = \phi(143) = (11 - 1) * (13 - 1) = 120$
- $gcd(e, \phi(n)) = 1$, $e = 23$
- public key: $e = 23$ und $n = 143$
- $d : 23 * d + k * 120 = 1 \rightarrow d = 47$

# RSA-Beispiel

Wir wollen die Zahlen 7 ($m_1$) und 3 ($m_2$) verschlüsseln:

- $c_1 \equiv 7^{23} \bmod 143 \rightarrow c_1 = 2$
- $c_2 \equiv 3^{23} \bmod 143 \rightarrow c_2 = 126$
- $m_1 * m_2 = 7 * 3 = 21,\ c_3 \equiv 21^{23} \bmod 143 \rightarrow c_3 = 109$
- $E(m_1) * E(m_2) = c_1 * c_2 \bmod \mathrm{n} = 2 * 126 \bmod \mathrm{n} = 109$
- $E(m_1 * m_2) = E(7 * 3) = E(21) = c_3 = 109$

Entschlüsselung: $109^{d} mod\ n = 109^{47} mod\ 143 = 21$

# Paillier-Verfahren

Wiederholung:

- $(\mathbb{Z}/n\mathbb{Z})$ Restklassenring: Zahlen mit gleichem Rest bei Division durch n werden zu Restklassen mod n zusammengefasst. Diese Restklassen bilden einen Ring.

  Beispiel: $(\mathbb{Z}/3\mathbb{Z}) = \{\mathbf{0}, \mathbf{1}, \mathbf{2}\}$
  $\mathbf{0} := \{...; -18; ...; -3; 0; 3; 6; ...\}$ d.h. die durch 3 teilbaren Zahlen
  $\mathbf{1} := \{...; -7; ...; 1; 4; ...\}$ d.h. Divisionsrest ist 1
  $\mathbf{2} := \{...; -8; ...; 2; 5; ...\}$ d.h. Divisionsrest ist 2

- $(\mathbb{Z}/n\mathbb{Z})^*$ prime Restklassengruppe: Wenn für eine Restklasse a gilt: $ggT(a, n) = 1$,dann heißt diese Klasse prime Restklasse mod n. Die Gruppe all dieser Restklassen heißt prime Restklassengruppe.

# Paillier-Verfahren

Vereinfachte Generierung des Schlüsselpaares:

- Wähle Sicherheitsparameter k
- Wähle $p, q \in \mathbb{P}$ so, dass sie die gleiche Länge in Bits haben (Länge =k)
- Man definiert ein $g = n + 1$
- public key = (n,g)
- Man definiert $\lambda = \phi(n) = (p-1)(q-1)$
- Man definiert $L(x) = \frac{x-1}{n}$

# Paillier-Verfahren

Verschlüsselung einer Nachricht $m \in (\mathbb{Z}/n\mathbb{Z})$:

- Wähle zufällig $r \in (\mathbb{Z}/n\mathbb{Z})^*$
- $c = g^m * r^n mod\ n^2$

Entschlüsselung für ein $c \in (\mathbb{Z}/n^2\mathbb{Z})^*$:
$m = L(c^\lambda mod\ n^2) * \lambda^{-1} mod\ n$
$\lambda^{-1}$ ist das multiplikative Inverse zu $\lambda\ mod\ n$

# Paillier-Verfahren

Das Verfahren ist homomorph bezüglich der Addition:
Durch Multiplikation von zwei Schlüsseltexten werden die verschlüsselten Klartexte addiert:

- $c_1 * c_2 \ mod \ n = (g^{m_1} * r_1^n) * (g^{m_2} * r_2^n) \ mod \ n = g^{m_1+m_2} * (r_1 * r_2)^n \ mod \ n = g^{m_1+m_2} * r'^n \ mod \ n$

- $D(c_1 * c_2 \ mod \ n^2) = m_1 + m_2 \ mod \ n$

# Paillier-Verfahren-Beispiel

- $k = 5$
- $p = 17$, binär: $p = 10001$
- $q = 19$, binär: $q = 10011$
- $n = 17 * 19 = 323$
- $g = n + 1 = 324$
- $\lambda = (p - 1) * (q - 1) = 16 * 18 = 288$
- $\lambda^{-1} : 288 * \lambda^{-1} + 323 * b = 1 \Rightarrow \lambda^{-1} = 203$

# Paillier-Verfahren-Beispiel

Wir wollen 7 und 3 verschlüsseln $(7, 3 \in (\mathbb{Z}/323\mathbb{Z})$:

- $r_1 \in (\mathbb{Z}/n\mathbb{Z})^* = 4$, da $ggT(323, 4) = 1$
- $r_2 \in (\mathbb{Z}/n\mathbb{Z})^* = 9$, da $ggT(323, 9) = 1$
- $c_1 = 324^7 * 4^{323} mod 323^2 = 14616$
- $c_2 = 324^3 * 9^{323} mod 323^2 = 54262$

- $c_1 * c_2 = 793093392$
- $E(m_1 + m_2) = E(10) = 324^{10} * 36^{323} mod\ 323^2 = 88663$
- $793093392\ mod\ 323 = 88663\ mod\ 323$
- $161 = 161$

- Entschlüsselung:
  $D(c_1 * c_2\ mod\ n^2) = D(793093392\ mod\ 323^2) = D(88663) =$
  $L(88663^{288}\ mod\ 323^2) * 203\ mod\ 323 = L(95609) \times$
  $203 mod\ 323 = 296 * 203\ mod\ 323 = 10 = 7 + 3$

# Vollhomomorphe Verschlüsselung

Ein Verschlüsselungsverfahren mit $E : R \to R'$ (R,R' Ringe) heißt voll homomorph, wenn gilt: $\forall m_1, m_2 \in R$:

- $E(m_1 +_r m_2) = E(m_1) +_{r'} E(m_2)$
- $E(m_1 *_r m_2) = E(m_1) *_{r'} E(m_2)$

D.h. es ist ein Kryptosystem, das Addition und Multiplikation unterstützt.

# Schwierigkeiten der vollhomomorphen Verschlüsselung

- Craig Gentry hat das erste voll homomorphe Kryptosystem 2009 vorgestellt. Dieses Verfahren basiert auf Problemen der Gittertheorie.
- Einwegfunktion gewährleistet Sicherheit des Kryptosystems $\rightarrow$ Gittertheorie liefert Probleme, die $\aleph P$-vollständig sind
- Die vorhandenen Verfahren für Vollhomomorphe Verschlüsselung sind zu langsam und ineffizient für Anwendungen

# Anforderung an ein E-Voting System

- **Korrektheit:** Eingegangene Stimmen nicht verändern oder löschen, ungültige nicht zählen
- **Demokratisch:** Nur berechtigte Wähler dürfen Stimme abgeben, jeder nur einmal
- **Vertraulich:** Niemand soll feststellen können, wie ein Wähler abgestimmt hat
- **Überprüfbarkeit:** Jeder Wähler kann selbständig prüfen, ob die eigene Stimme korrekt gezählt wurde
- **Gerechtigkeit:** keine Zwischenergebnisse verlautbaren, bevor die Abgabefrist abläuft

# E-Voting mit Paillier

- Wahlleiter berechnet $n = p * q$ (öffentlich)
- Einfache Variante: Wähler i wählt $v_i = 0$ für NEIN, $v_i = 1$ für JA
- $c_i = g^{v_i} * r_i^n mod\ n^2 \rightarrow$ öffentlich
- $c := \prod_{i=1}^{n} c_i$
- Wahlleiter erhält c, entschlüsselt es und veröffentlicht das Ergebnis
- Es gilt: $D(c) = \sum_{i=1}^{n} v_i$

# E-Voting mit Paillier

Dadurch ist erfüllt:

- Leiter kennt keine der einzelnen $c_i$'s
- Kein Wähler kennt die $v_i$'s der anderen Wähler
- Wie c berechnet wurde ist öffentlich verifizierbar

# Cloud Computing

Es ist manchmal von Vorteil, Ressourcen von einem Cloud Provider zu kaufen, als selbst ein Datencenter aufzubauen. Immer mehr nützen Cloud Computing um Daten zu analysieren. Hat man es jedoch mit privaten Daten zu tun, mit denen man Berechnungen anstellen will, stellt sich die Frage:

Wie sehr vertraut man dem Cloud Provider?

Wie kann man am Besten die Vorteile der Cloud nützen, aber auch die Sicherheit der Daten nicht gefährden?

Es ist ein zu großes Sicherheitsrisikio einem öffentlichen Cloud Provider, wie z.B. Google, Zugriff zu unverschlüsselten Daten zu gewährleisten

# Cloud Computing und Homomorphe Verschlüsselung

Das Prinzip:

- Alice will eine Datei $x \in \{0,1\}^m$ auf Bob's Server speichern
- Sie schickt ihm $E(x_1)...E(x_m)$
- Nun möchte sie Berechnungen auf dieser Datei ausführen
- Sie könnte natürlich alles wieder downloaden und selbst die Berechnungen durchführen
- Hätte sie aber die Möglichkeit große Berechnungen auf ihrem Rechner zu erledigen, hätte sie sich wahrscheinlich nicht für Cloud Computing entschieden
- Sie bittet Bob diese Aktion auf der verschlüsselten Datei durchzuführen und ihr dann das (verschlüsselte) Ergebnis zu senden
- Sie kann das Resultat problemlos entschlüsseln.

Wären die Daten "konventionell" verschlüsselt, wäre es dem Provider nicht möglich, auf ihnen Operationen auszuführen, ohne ein falsches Ergebnis zu erhalten.

# Cloud Computing und Homomorphe Verschlüsselung - ein Beispiel mit RSA

- $p = 13$, $q = 23$, $n = 299$, $\phi(n) = 264$, $e = 5$, $d = 53$
- Firma XYZ hat eine Menge von Daten, z.B. jeweils 5 Produkte an 10 Kunden verkauft. $\{5, 10\}$
- XYZ verschlüsselt die Daten: $E(5) = c_1 \equiv 5^5 \mod 299 = 135$, $E(10) = c_2 \equiv 10^5 \mod 299 = 134$
- XYZ schickt die Menge der verschlüsselten Daten (135,134) an die Cloud
- ein paar Monate später, will die Firma wissen, wie viele Produkte sie insgesamt verkauft hat (5*10)
- der Cloud Provider berechnet 135*134 und schickt das Ergebnis (18090) an XYZ
- diese entschlüsselt die Zahl $m = 18090^{53} mod\ 299 = 50 = 10 * 5$