(Teil-)Homomorphe Verschlüsselung

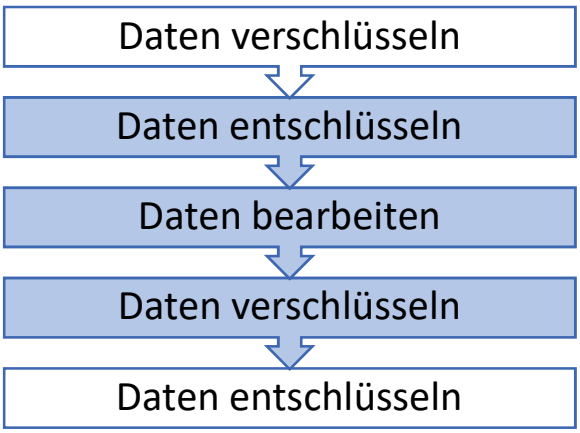
Motivation

Wozu homomorphe Verschlüsselung?

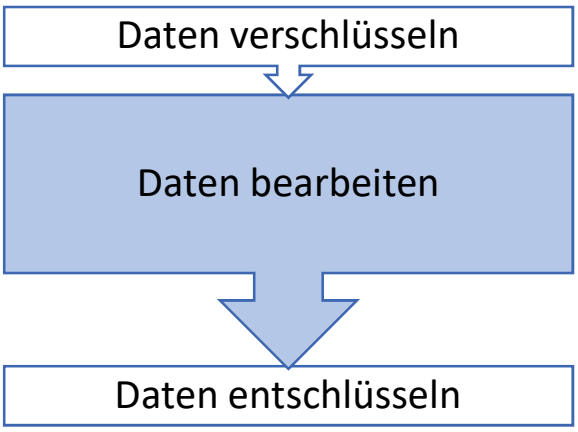
Das Ziel von Verschlüsselung ist grundsätzlich die Daten vor dem Zugriff von anderen zu schützen. Will man aber Berechnungen auf den verschlüsselten Daten durchführen muss man diese zuerst entschlüsseln, danach die Berechnung durchführen und danach wieder verschlüsseln. Bei diesem Vorgehen gibt es mehrere Möglichkeiten auf die nicht-verschlüsselten Daten zuzugreifen. Will man die verschlüsselten Daten auslagern müssen die Daten zusätzlich auch noch für jede Berechnung übertragen werden.

Mit homomorpher Verschlüsselung ist es möglich, die Berechnungen direkt auf die verschlüsselten Daten auszuführen.

ohne homomorpher Verschlüsselung



mit homomorpher Verschlüsselung



Wo wird homomorphe Verschlüsselung verwendet?

Cloud Computing

- Outsourcing der Daten auf ein Cloudservice
- Beispiel Analyse medizinischer Daten

e-Voting

- Stimmen verschlüsseln
- Berechnung des Wahlergebnisses mit den verschlüsselten Daten
- dazu später mehr

Was bedeutet nun homomorph?

Dazu wiederholen wir kurz die Definitionen von Gruppen und Ringen.

Definition Gruppe

Eine Gruppe ist ein Paar $(G, \circ)$. G ist eine Menge und $\circ$ eine zweistellige Verknüpfung

$\circ: G \times G \to G$ und $(a,b) \mapsto a \circ b$. mit den folgenden Eigenschaften:

- *Assoziativität* $\forall\, a,b,c \in G : (a \circ b) \circ c = a \circ (b \circ c)$
- *neutrales Element:* $\exists\, e \in G\ \forall\, a \in G \quad : a \circ e = e \circ a = a$
- *inverses Element:* $\forall\, a \in G\ \exists\, a^{-1} \in G : a \circ a^{-1} = a^{-1} \circ a = e$
- Eine Gruppe heißt abelsch, wenn das Kommutativgesetz gilt:

$\forall\, a, b \in G : a \circ b = b \circ a$

Definition Ring

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

Dabei muss gelten:

- (R,+) ist eine abelsche Gruppe.
- R ist bezüglich $*$ abgeschlossen
- Bezüglich $*$ gilt die Assoziativität
- Die Distributivgesetze gelten

Definition Homomorphismus

Seien $(G, \circ)$ und $(H, \diamond)$ Gruppen, dann heißt die Abbildung $f: G \longrightarrow F$ **Gruppenhomomorphismus**, wenn $\forall\, a,b \in G$ gilt:

$$f(a \circ b) = f(a) \diamond f(b)$$

Seien $(R, +_r, *_r)$ und $(S, +_s, *_s)$ Ringe, dann heißt die Abbildung $\varphi: R \longrightarrow S$ **Ringhomomorphismus**, wenn $\forall\, a, b \in R$ gilt:

$$\varphi(a +_r b) = \varphi(a) +_s \varphi(b)$$
$$\varphi(a *_r b) = \varphi(a) *_s \varphi(b)$$

Historische Entwicklung

1978 erster Versuch von Rivest, Adleman und Dertouzes für teilhomomorphe Verschlüsselung. Bereits damals setzte man sich vollhomomorphe Verschlüsselung zum Ziel.

1987 wurde das erste teilhomomorphe Verfahren durch Brickell und Yacobi entschlüsselt.

2009 hat Gentry bewiesen, dass vollhomomorphe Verschlüsselung (theoretisch) möglich ist. Zu diesem Zeitpunkt gab es aber noch keine praktische Anwendung.

Varianten

Es gibt verschiedene Varianten der homomorphen Verschlüsselung

- additiv homomorph: hier ist nur die Addition homomorph
- multiplikativ homomorph: hier ist nur die Multiplikation homomoprh
- hybrid-homomorph: die Verknüpfung von zwei teilhomomorphen Verschlüsselungen
- voll-homomorph: additiv und multiplikativ homomorph

Beispiel additiv-homomorph

Hans ist auf Urlaub als er krank wird und zum Arzt geht. Dieser nimmt ihm Blut ab und untersucht dieses auf einen bestimmten Wert. Da die Untersuchung nicht gut ausfiel bekommt er ein Medikament gegen die Schmerzen und wird zur Beobachtung ins Krankenhaus überwiesen. Dort bekommt er gleich noch eine Dosis des Medikaments verschrieben und wird zur Beobachtung dabehalten.

Am nächsten Tag bekommt er noch eine doppelte Dosis des Schmerzmittels für den Tag und die Heimreise. Als er zuhause in der Nacht aufwacht und vor Schmerzen nicht mehr einschlafen kann geht er ins Krankenhaus in seinem Wohnort, dort muss nun geprüft werden ob er die tägliche Dosis des Wirkstoffs noch nicht überschritten hat. Die Daten zur Medikation sind verschlüsselt in der online-Patientenakte gespeichert.

Durch die homomorphe Verschlüsselung kann der Wert direkt abgefragt werden.

Zur Veranschaulichung hier mit einer sehr einfachen Verschlüsselung der Medikation:

Verschlüsselung E(x) = x*k , k=3 und die dazugehörige Entschlüsselung D(x) = x/k

unverschlüsselte Daten	1	2	3	=	6
	⇩	⇩	⇩		⇩
verschlüsselte Daten	3	6	9	=	18

alternative additive Verschlüsselung $(G,+) \rightarrow (G,+)$ $x \rightarrow e^x$ $e^{x1}\, e^{x2} = e^{(x1+x2)}$

Beispiel multiplikativ-homomorph

$(G, *) \longrightarrow (G, *)$ $x \longrightarrow x^2$

$x_1^a\, x_2^a = (x_1\, x_2)^a$

Verschlüsselung $E(x) = x^2$
Entschlüsselung $D(x) = \sqrt{x}$

$2 * 3 * 7 = 42$

$(2*3*7)^2 = 42^2 = 1764$
$E(2) + E(3) + E(7) = 2^2 * 3^2 * 7^2 = 4 * 9 * 49 = 1764$

$D(36) = \sqrt{1764} = 42$

In diesen Beispielen wurden aufgrund der Anschaulichkeit symmetrische Verschlüsselungen gewählt.

RSA

Das klassisches RSA besitzt teilhomomorphe Eigenschaften im Bezug auf die Multiplikation von Ciphertexten zur Multiplikation (mod N) der Plaintexte. Beweis hierfür folgt aus den Rechenregeln der Restklassensystemen

$$\text{(konkret: a (mod x) * b (mod x) = a*b (mod x)}$$

Beispiel mit 2 Ciphertexten:

$$c_1 * c_2 = ((m_1^e \pmod N) * (m_2^e \pmod N)) \pmod N$$

Dies ergibt nach Umformung

$$c' = (m_1 m_2)^e \pmod N$$

Wird dies nun mit dem zugehörigen privaten Schlüssel entschlüsselt, ergibt sich

$$((m_1 m_2)^e \pmod N)^d \pmod N = ((m_1 m_2)^{e*d} \pmod N =$$

$$(m_1 m_2) \pmod N$$

Padded RSA – OAEP

Optimal **A**symmetric **E**ncryption **P**adding ist eine der Standardisierten Verfahren um RSA zu „padden“. Hierbei wird die Nachricht bevor sich mit RSA verschlüsselt wird mit OAEP „verschlüsselt“.

RSA – OAEP besitzt allerdings keine homomorphen Eingenschaften.

Hierbei wird eine zu sendende Nachricht auf eine fixe Länge gebracht (gegebenenfalls wird mit 0 erweitert) und ein Sicherungsblock mit fester Länge mit Zufallszahlen gefüllt. Zudem werden 2 Kryptographische Hashfunktionen benötigt (d.h. Hashfunktionen, die one-Way sind, und keine zwei Werte auf den selben Wert hashen). Diese müssen als Parameter einen Wert der Länge des Messageblocks oder des Sicherungsblock nehmen, und einen Wert mit Länge des anderen Blocks wiedergeben.

Verschlüsselung

Die Verschlüsselung läuft wie folgt:

1. m vorbereiten, r generieren
2. r auf G(r) hashen
3. X aus m $\oplus$ G(r) berechnen
4. X auf H(X) hashen
5. Y aus r $\oplus$ H(X) berechnen
6. X, Y zu m' verknüpfen

Entschlüsselung

Entschlüsselung läuft dann mit:

1. X, Y aus m' auslesen
2. X auf H(X) hashen
3. r aus Y $\oplus$ H(X) berechnen
4. r auf G(r) hashen
5. m aus X $\oplus$ G(r) berechnen

Goldwasser-Micali

Das Goldwasser-Micali Verfahren verschlüsselt einzelne Bits.

Das Verfahren basiert stark auf der Prüfung ob eine Zahl ein quadratischer Rest ist.

Dies heißt konkret:

Wir betrachten eine Zahl z sowie eine Zahl p, wo ggT(z,p) = 1

falls es eine Zahl x gibt wo

$z = x^2 \pmod p$

dann ist z (mod p) ein quadratischer Rest.

Gibt es keine solche Zahl x, ist z (mod p) ein quadratischer Nichtrest.

Die Berechnung ist für p Prim simple. In diesem Fall muss lediglich das Legendre-Symbol berechnet werden ().

Für allgemeine p gibt es kaum effiziente Möglichkeiten dies zu bestimmen, es sei denn die Faktorisierung von P ist bekannt. Falls p wie bei Goldwasser-Micali nur aus teilerfremden Primfaktoren besteht, kann für die Prüfung ob eine Zahl zu p quadratischem Rest ist, vereinfacht werden, indem geprüft wird ob die Zahl zu allen Faktoren quadratischer Rest ist.

Schlüsselgenerierung

Für die Schlüsselgenerierung generieren wir p,q Prim (wie bei RSA) und das Produkt N = pq.

Dass wählen wir ein x das (mod N) quadratischer Rest ist.

(p,q) bilden dann den privaten Schlüssel, (x,N) den öffentlichen Schlüssel.

Ver- / Entschlüsselung

Der Ciphertext c wird dann mit der Formel berechnet

$$c = r^2 * x^m \ (mod\ N)$$

Bei der Entschlüsselung wird nun berechnet ob c ein quadratischer Rest (m = 0) oder ein quadratischer Nichtrest (m = 1) ist. Dementsprechend wird der Wert 0 oder 1 zurückgegeben.

Teilhomomorphie

Wir betrachten nun wieder die Multiplikation $(mod\ N)$ zweier Chiffrate c_1, c_2, die mit dem öffentlichen Schlüssel (x, N) erzeugt wurden:

$$c1 * c2 \ = \left(r_1^2 * xm1(mod\ N)\right) * \left(r_2^2 * xm2(mod\ N)\right) = (r_1 r_2)^2 * xm1 + m2 \ \ (mod\ N)$$

Sollte in diesem Beispiel $m1 \ = \ m2 \ = \ 1$ sein, lässt sich folgendes zeigen.

$$(r_1 r_2)^2 * x^{1+1} \ \ (mod\ N) = (r_1 r_2 x)^2 * x^0 \ \ (mod\ N)$$

Anhand dieser Veranschaulichung lässt sich nun leicht nachvollziehen, dass die Multiplikation der Chiffrierte eine Addition Modulo 2 (XOR) der Plaintexte entspricht.

Jedes Vielfache von 2 in der Summe der (m_i) wird zu einem quadratischen Rest.

Paillier

Erfunden von und benannt nach Pascal Paillier im Jahr 1999. Es ist ein additives homomorphes Verschlüsselungsverfahren.

Anwendungen

- E-Voting

  Nachdem alle Wahlberechtigten ihre Stimmen verschlüsselt übermittelt haben, wird das Ergebnis aus diesen berechnet und dann entschlüsselt. Für die Errechnung des Ergebnisses wird der Private-Key nicht benötigt.

- Zero-Knowledge Beweis

  Ist eine Methode, mit der eine Partei einer anderen Partei beweisen kann, dass sie über Wissen verfügen ohne Informationen über dieses Wissen weiter zu geben.

  Beispiel: 2 Bälle mit unterschiedlichen Farben.

Paillier KeyGen

Man wählt zwei ausreichend große Primzahlen p und q. Ausreichend groß entspricht 1024 Bit, also Zahlen mit je 309 Ziffern.

Danach berechnet man n. n ist das Produkt von p und q. Danach berechnet man l als das kleinste gemeinsame Vielfache von (p-1) und (q-1). Vereinfacht kann man auch l = (p-1)(q-1) setzen.

Man wählt dann noch ein zufälliges g aus dem Restklassenring modulo n^2, oder vereinfacht setzt man g auf n+1.

Außerdem benötigt man noch das multiplikative Inverse m zu l modulo n.

Daraus ergibt sich der Public Key (n,g) und der Private Key (l,m)

Paillier Verschlüsselung

Zur Berechnung des Ciphertextes mit dem Public Key (n,g) braucht man noch eine Zufallszahl r, für die gilt: ggT(r,n) = 1 und 0 < r < n.

Dann ergibt sich der Ciphertext c einer Nachricht m mit der Formel:

$$c = g^m r^n \; mod \; n^2$$

Paillier Entschlüsselung

Man berechnet die Nachricht m aus dem Ciphertext c mit Hilfe des Private Keys (l,m) (und des Public Keys (n,g) natürlich) durch die Formel

$$m = \frac{\left(c^\lambda \; mod \; n^2\right) \; - \; 1}{n} \mu \; mod \; n$$

Paillier Teilhomorphie

Die Verschlüsselung ist additiv homomorph

Eine Multiplikation von zwei verschlüsselten Werten entspricht der Addition der unverschlüsselten Werte. $c_1 * c_2 \;\equiv_{n^2}\; (g^{m_1} r_1^n) * (g^{m_2} r_2^n) \;\equiv_{n^2}\; g^{m_1+m_2} \, (r_1 r_2)^n$

Da der $ggT(r_1 r_2, \, n) = 1$ ist, ist $c_1 c_2$ ein gültiger Ciphertext von $m_1 + m_2$.

Übersicht über Teilhomomorphe Algorithmen

Algorithmus	Homomorphie Eigenschaft
RSA	multiplikativ homomorph
Padded RSA (OAEP)	nicht homomorph
Goldwasser Micali	additiv homomorph
Paillier	additiv homomorph

Vollhomomorphe Verschlüsselungen

Eine Funktion behält die Ringstruktur von $(\mathrm{R},\ +,\ *)$ bei. Also $(\mathrm{f}(\mathrm{R}),\ +,\ *)$ ist immer noch ein Ring, wenn f vollhomomorph ist.

Das heißt bei einer Vollhomomorphen Verschlüsselung können Additionen und Multiplikationen durchgeführt werden.

Gentry‘s Algorithmus (2009) war der erste vollhomomorphe Verschlüsselungsalgorithmus, mit Verwendung von Zahlengittern. Seine Laufzeitkomplexität ist $\mathrm{O}(\lambda^{10})$.

Aktuelle Laufzeiten

Energieverbrauch Profilklassifikation	**< 1 Sekunde**
Verschiedene medizinische Diagnosen	< 2 Minuten
Gen-basierte Diagnosen	< 10 Minuten
Lauflängenkodierung (bei 48 Kernen) Zur Bild/Videokompression.	Ca 30 Minuten

Komplexität basiert auf dem Sicherheitslevel der Daten, der Komplexität der Daten und der Optimierung.

Probleme

Die Wahl geeigneter Parameter ist schwer. Außerdem sind bisherige Implementierungen nicht alltagstauglich, zum einen wegen der Geschwindigkeit der Algorithmen. Zum anderen verrauschen die Daten. Bei Gentry zum Beispiel muss nach spätestens 30 Operationen eine Bereinigung durchgeführt werden, damit die Daten korrekt und lesbar bleiben.

Auch in der Sicherheit gibt es einen Angriffspunkt. Die Verschlüsselungen sind mit Chosen-Ciphertext-Attacken angreifbar.

Hybrid-Homomorph

Bei einem hybrid-homomorphen Verschlüsselungsalgorithmus, werden zwei verschiedene teilhomomorphe Verschlüsselungsalgorithmen verwendet. Zum einen eine additive homomorphe Verschlüsselung und eine multiplikativ homomorphe Verschlüsselung. In dem von uns gewählten Verfahren wurde Goldwasser-Micali als additive Variante gewählt und RSA als multiplikative Variante.

Ablauf

1. Erstellen der Keys für G. Micali
2. Verschlüsseln mittels G. Micali
3. Erstellen der Keys für RSA
4. Verschlüsseln mittels RSA
5. Durchführen der Operationen
6. Entschlüsseln mittels RSA
7. Entschlüsseln mittels G. Micali

Aufpassen muss man hier nur, dass wenn zuerst mit Goldwasser-Micali verschlüsselt wird und dann mit RSA. Dann muss bei der Entschlüsselung zuerst mit RSA entschlüsselt und dann mit Goldwasser-Micali entschlüsselt werden.

Vor- und Nachteile

Nachteile

- Sehr hohe Laufzeiten
- Große Rechnerkapazitäten werden benötigt
- Bei der Anwendung, ist die gültige Rechtsprechung, abhängig vom Standort der Server

Vorteile

- Gewinn an Datenschutz
- Mobilität
- Outsourcing

Anwendungen

Cloud Computing

Homomorphe Verschlüsselung findet sich in Anwendungen wie Cloud Computing und E-Voting wieder.

Bei Cloud Computing erhofft man sich durch Homomorphe Verschlüsselungsverfahren einen Gewinn an Datenschutz. Sicherheit dient als höchste Priorität, damit Unternehmen einen Wechsel zur Cloud in Betracht ziehen.

Verschlüsselung der Daten ist insofern notwendig, da meist unklar ist, in welchem Land die Daten gespeichert werden. Daher ist Datenverschlüsselung aus Datenschutzgründen unerlässlich.

Durch das Verfahren werden verschlüsselte Daten in der Cloud abgelegt, wodurch Datenbanken beispielsweise diese Daten zuverlässig speichern und Berechnungen auf den gespeicherten Daten ausführen können. Die Auswertung auf verschlüsselte Daten führt zu einem verschlüsselten Ergebnis, das der anfragende Benutzer erhält und entschlüsselt. Dadurch kann weder der Cloud Anbieter selbst, noch potenzielle Angreifer auf die Cloud, die gespeicherten Daten noch das Ergebnis entschlüsseln.

Partiell Homomorphe Verschlüsselung findet sich bereits in Anwendungen wieder. Hingegen Voll Homomorphe Verschlüsselungsverfahren aufgrund der Komplexität und der Rechenintensität noch nicht. Obwohl die Verfahren in der Theorie immer weiter verbessert werden, sind sie in der Praxis noch schwer anzuwenden aufgrund der exponentiellen Vergrößerung des Schlüsseltextes gegenüber dem Klartext und der langen Berechnungsdauer.

E-Voting

Anforderungen an elektronische Wahlen sind vor allem die Manipulationssicherheit und das Wahlgeheimnis. Stimmabgaben dürfen nicht manipuliert werden und die Auszählung soll überprüfbar sein. Zudem sollen die Wahlen geheim sein.

Dass jeder Internetnutzer versuchen kann, in die Wahl einzugreifen, sowohl das nur Wahlberechtigte die Möglichkeit bekommen sollen, eine gültige Stimme abzugeben, stellt die Entwickler vor großen Herausforderungen.

Protokoll

In unserem Protokoll erhält der Wähler zusammen mit dem Wahlbescheid die erforderlichen Zugangsdaten, bestehend aus Wähler-ID und einem Passwort. Damit logt sich der Wähler auf der Wahlwebsite ein, wählt einen Kandidaten aus und schickt seine Stimme ab.

Jedoch kann das Problem auftreten, dass der Angreifer die Zugangsdaten stiehlt (beispielsweise aus dem Briefkasten). Daher erfolgt eine zusätzliche Identitätsprüfung mittels der eID (elektronischer Personalausweis) um sich als Wahlberechtigter zu authentifizieren.

Das Protokoll ist einfach verwendbar, aber noch nicht sicher, da sich der Angreifer zwischen Wähler und digitaler Wahlurne schalten und die Wahlentscheidung mitschneiden kann.

Dies lässt sich allerdings mit asymmetrischer Verschlüsselung lösen, sodass das Mitschneiden durch den Angreifer nicht länger funktioniert.

Probleme

Ein weiteres Problem ist der böse Wahlleiter. Denn noch ist unsere Stimme nicht komplett geheim, weil Wahlurnenserver den Zusammenhang zwischen Wähler und dem jeweiligen Geheimtext kennt. Wird der Server zur Entschlüsselung genutzter, kann der Wahlleiter oder ein Admin einen

Zusammenhang herstellen. Um das Problem zu beheben, erweitert man das Protokoll mittels Homomorpher Verschlüsselung. Nun lassen sich einfache mathematische Operationen auf den verschlüsselten Stimmen ausführen, die den gleichen Operationen auf dem Klartext entsprechen, ohne sie dafür entschlüsseln zu müssen. Man addiert die Stimmen im verschlüsselten Zustand. Durch die Addition entsteht ein Geheimtext, der das Gesamtergebnis der Wahl enthält. Dies verhindert allerdings nicht, dass einzelne Stimmen entschlüsselt werden, indem jemand der geheimen Schlüssel anwendet, um das Zwischenergebnis zu entschlüsseln. Es muss also gewährleistet werden, dass der geheime Schlüssel nicht vor dem Wählende eingesetzt wird. Das Prinzip hierbei ist Secret Sharing. Dabei wird der private Schlüssel in t Teilen erzeugt. Um nun einen gültigen Schlüssel zusammenzusetzen werden allerdings nur n Teile benötigt, wobei $n < t$.

Zusätzlich gibt es noch weitere Überprüfungsmöglichkeiten für den Wähler, sodass dieser weiß, dass seine verschlüsselte Stimme auf dem Wahlurnenserver angekommen ist und in das Endergebnis integriert wurde.

Dies zeigt, dass End-to-End verifizierbare Wahlen über das Internet möglich sind. Ein Vorteil ist vor allem die hohe Sicherheit und dass weiterhin an besseren und sicheren, Protokollen geforscht wird.

Herausforderungen sind nach wie vor die hohe Komplexität. Kryptografie und Zero-Knowledge-Proof erfordern ein tief gehendes Vorwissen, welches die Wähler nicht haben. Der Wähler muss sich daher auf unabhängige Experten verlassen können, welche die Wahl beobachten und die Software kontrollieren.