Homomorphe Verschlüsselung

Ein Beitrag des AK-Krypto des BvD – Autor: Riko Pieper

Der (ehemalige) BfDI Prof. Kelber hat in seinem „Abschiedsbrief“[i] vom Juli 2024 darauf hingewiesen, dass die „Privacy Enhancing Technologies“ (PET) und hier explizit die „homomorphe Verschlüsselung“ Themen sind, bei denen es in den letzten Jahren vielversprechende Fortschritte gab, deren weitere Entwicklung man im Auge behalten muss. Die Nachfolgerin von Prof. Kelber - Frau Prof. Dr. Specht-Riemenschneider - sieht das übrigens genauso, was seitens der Pressestelle der BfDI bestätigt wurde.

Wer, wenn nicht der AK-Krypto des BvD sollte sich hier besonders angesprochen fühlen. In diesem Artikel soll das Konzept der homomorphen Verschlüsselung anhand von einfachen Beispielen so erläutert werden, dass jeder Datenschützer[ii] über die Grundlagen informiert ist und weiß, worum es hier geht, welche praktischen Anwendungsfälle es gibt und welche Probleme (noch) existieren; denn gäbe es keine Probleme, dann wäre diese Technologie längst überall im Einsatz, wo sie sinnvoll eingesetzt werden kann.

1. Worum es geht – der Anwendungsfall

Bei der Verschlüsselung geht es allgemein darum, dass Daten so „verändert“ (verschlüsselt) werden, dass sie von Unbefugten nicht mehr interpretiert werden können. Nur wer den passenden Schlüssel kennt, kann die Daten wieder entschlüsseln. Zum Beispiel Festplatten oder andere Datenträger können so verschlüsselt werden. Wenn sich ein Unbefugter so einen Datenträger aneignet, kann er damit nichts anfangen, solange er den Schlüssel nicht kennt. Für diese sogenannte „symmetrische“[iii] Verschlüsselung gibt es einen Standard – den AES[iv], der nach dem Stand der Technik als „sicher“ gilt.

Bei der Übermittlung von verschlüsselten Daten kommt ein weiteres Problem dazu – man muss sicherstellen, dass der Empfänger (und nur der) den passenden Entschlüsselungsschlüssel hat. Dieses Problem wurde mit der „Public Key Kryptographie“ – der Verschlüsselung mit Hilfe von öffentlichen und privaten Schlüsseln – gelöst, die daher auch als „asymmetrische Verschlüsselung“ bezeichnet wird. Dabei wird mit dem öffentlichen Schlüssel des Empfängers (den dieser öffentlich bekannt gibt) verschlüsselt. Entschlüsselt werden kann die verschlüsselte Nachricht nur vom Empfänger, denn er hat dieses Schlüsselpaar ursprünglich generiert und er hat den zweiten (privaten) Entschlüsselungsschlüssel niemandem mitgeteilt. Vorstellen kann man sich das auch anhand einer mit einem Vorhängeschloss (entspricht dem öffentlichen Schlüssel) an einer derart (verschlüsselten) Kiste, deren Vorhängeschlösser überall erhältlich sind. Den dazu passenden (privaten) Schlüssel zum Öffnen der Schlösser hat aber nur der (berechtigte) Empfänger.

Dieses Gedankenexperiment kann man an vielen Stellen nachlesen. Besonders zu empfehlen ist dafür das Buch[v] „Geheime Botschaften. Die Kunst der Verschlüsselung von der Antike bis in die Zeiten des Internet.“ von Simon Singh, in dem auch viele andere Verschlüsselungsverfahren (und dazugehörige historische Hintergründe) sehr anschaulich beschrieben sind. Das diesbezüglich bekannteste und verbreitetste Verfahren ist das RSA-Verfahren[vi], das z. B. bei der E-Mail-Verschlüsselung (sowohl bei S/MIME als auch bei PGP) genutzt wird. Das im Buch

erläuterte Verfahren wird anschaulich einschließlich eines einfachen Rechenbeispiels erklärt und soll hier daher nicht weiter vertieft, sondern als bekannt vorausgesetzt werden.

Es stellt sich nun die Frage, ob es bezüglich der Verschlüsselung noch offene Probleme gibt, denn die „eigentliche“ Verschlüsselung ist mit dem AES-Verfahren gelöst und das Problem des Schlüsselaustausches ist mit dem RSA-Verfahren auch gelöst.

Die wenig überraschende Antwort lautet: Ja!

Die Zeiten, in denen die Unternehmen ihre gesamte Datenverarbeitung auf eigenen Rechnern und mit eigenem Personal durchführen konnten, gehören der Vergangenheit an. In der heutigen Realität wird immer mehr in die „Cloud“ ausgelagert und hier stellen sich folgende Fragen: Kann man dem Cloud-Anbieter wirklich trauen? Kann man darauf vertrauen, dass die eigenen Daten vom Cloud-Anbieter nicht weitergegeben werden?

Dieser Text soll kein politischer sein, sondern einer des AK-Krypto des BvD. Deshalb sollen hier keine Spekulationen folgen, sondern nur ein paar Fakten, so dass sich jeder die oben gestellten Fragen selbst beantworten kann:

Die meisten/bekanntesten Cloud-Anbieter kommen aktuell aus den USA. Nehmen wir einen seriösen Anbieter an (etwas anderes soll gar nicht unterstellt oder in Betracht gezogen werden), der einem zusichert, dass er die Daten nur gemäß den vertraglich vereinbarten Festlegungen verarbeitet (typischer AV-Vertrag gemäß Art. 28 DSGVO). Nehmen wir weiterhin an, dass die Bedingungen nach Kap. 5 DSGVO erfüllt sind.

Was bedeuten die vertraglichen Zusagen, wenn die Cloud-Anbieter Gesetzen wie dem FISA, oder Cloud Act[vii] unterliegen. Den Unternehmen kann man glauben, dass sie die Daten nicht weitergeben wollen – das hilft im Zweifel aber nichts. Da hilft auch kein „Data Privacy Framework“ (DPF)[viii] oder die Frage, ob dieses das nächste EuGH-Urteil übersteht. Es geht auch nicht nur um Cloud-Anbieter aus den USA, sondern um alle Anbieter, die sich an ähnliche Vorgaben der jeweils eigenen Regierung halten müssen.

Es ist zwar schön, wenn man einem Cloud-Anbieter vertrauen **kann**, aber es ist nicht schön (aus Sicht der IT-Sicherheit und insbesondere aus Sicht des Datenschutzes), wenn man ihm vertrauen **muss**. Es geht also darum eine Lösung zu finden, bei der man **nicht darauf vertrauen muss**, dass der Cloud-Anbieter die Daten (ggf. auf Druck einer Regierung) nicht doch weitergibt.

Unter diesen Rahmenbedingungen kann eine Cloud nur zuverlässig genutzt werden, wenn die Daten ausschließlich verschlüsselt in die Cloud übergeben werden. Damit ist nicht nur die Übermittlung der Daten gemeint (das ist Stand der Technik), sondern auch die Speicherung der Daten in der Cloud. Die Verschlüsselung muss dabei vom Verantwortlichen selbst mit einem eigenen Schlüssel durchgeführt werden – nicht mit Schlüsseln oder Software des Cloud-Anbieters[ix]. Nur dann kann man sich sicher sein, dass die Daten vom Cloud-Anbieter nicht entschlüsselt herausgegeben werden. Das bedeutet aber (bei den bisher gängigen Verschlüsselungsverfahren), dass man die Cloud nur für die Speicherung und den späteren Abruf der (verschlüsselten) Daten nutzen kann. Eine „Verarbeitung“ der Daten, wie sie im typischen Anwendungsfall üblich ist, wäre damit nicht möglich.

Die Lösung:

Jetzt kommen wir zum Thema des Artikels – zur „homomorphen[x] Verschlüsselung“ – Homomorphic Encryption (HE).

Es geht also darum einen Weg zu finden, bei dem ein Cloud-Anbieter die ihm zur Verfügung gestellten (verschlüsselten) Daten verarbeiten kann – ohne, dass er die Daten entschlüsseln kann/muss. Das geht mit den bisher üblichen Verschlüsselungsverfahren nicht. So ganz stimmt die Aussage im letzten Satz nicht; denn schon Rivest (der für das „R“ im RSA-Verfahren steht) hatte entdeckt, dass die Multiplikation zweier mit dem RSA-Verfahren verschlüsselten Daten ein Ergebnis liefert, das bei der anschließenden Entschlüsselung tatsächlich dem Produkt der beiden ursprünglichen/unverschlüsselten Zahlen entspricht. Diese Eigenschaft des RSA-Verfahrens brachte ihn auf die Idee einer homomorphen Verschlüsselung, bei der nicht nur die Multiplikation, sondern alle algebraischen Operationen durchgeführt werden können und die anschließende Entschlüsselung des Ergebnisses auch den Wert liefert, der ohne Verschlüsselung herausgekommen wäre. Eine mathematische Lösung dazu fand er zwar nicht, aber die Idee war geboren und wurde von ihm bereits 1978, nur 2 Jahre nach Veröffentlichung des RSA-Algorithmus, in einem Papier „On Data Banks and Privacy Homomorphisms“[xi] als neues Ziel formuliert.

2. Einfache Rechenbeispiele

2.1 Berechnung eines Durchschnitts

Stellen wir uns eine Schulklasse vor, in der eine Deutscharbeit geschrieben wurde. Jeder Schüler kennt nur seine eigene Zensur (1-6). Die Deutschlehrerin kennt die Zensuren selbstverständlich auch, spricht aber nicht darüber.

Nun hat der Mathematiklehrer die Idee, mit der Klasse gemeinsam den Zensur-Durchschnitt dieser Deutscharbeit zu ermitteln. Dieser spezielle Mathematiklehrer ist gleichzeitig Datenschutzbeauftragter der Schule und hatte erst kürzlich eine DSgzS-Veranstaltung[xii] des BvD an der Schule organisiert. Daher ist es ihm auch wichtig den Zensur-Durchschnitt derart zu ermitteln, dass dabei keiner der Schüler seine Zensur preisgeben muss. Der Mathematiklehrer geht dabei folgendermaßen vor:

Er lässt die Schüler in einem Kreis aufstellen und bittet einen beliebigen Schüler, sich eine beliebige ganze Zahl[xiii] auszudenken und für sich zu behalten. Danach bittet er diesen Schüler, seine eigene Zensur zu dieser Zahl zu addieren und das Ergebnis seinem rechten Nachbarn (leise) ins Ohr zu flüstern. Danach soll der zweite Schüler seine Zensur zu dem ihm mitgeteilten Wert addieren u.s.w.

Nachdem der letzte Schüler im Kreis seine Zensur addiert und dann das Ergebnis dem ersten Schüler mitgeteilt hat, subtrahiert dieser (der erste, mit dem es angefangen hatte) seine gedachte – nur ihm selbst bekannte – Zahl und gibt das Ergebnis laut bekannt.

Beispiel: Es handelt sich um eine kleine Klasse mit 10 Schülern und folgenden Noten: 1, 1, 5, 4, 1, 1, 2, 1, 3, 1.

Der erste Schüler denkt sich die 15. Dann kommt man zu folgenden Werten, die berechnet und an den jeweils nächsten Schüler im Kreis weitergegeben werden:

1. Schüler berechnet 15+1=16
2. Schüler berechnet 16+1=17
3. Schüler berechnet 17+5=22

4. Schüler berechnet 22+4=26
5. Schüler berechnet 26+1=27
6. Schüler berechnet 27+1=28
7. Schüler berechnet 28+2=30
8. Schüler berechnet 30+1=31
9. Schüler berechnet 31+3=34
10. Schüler berechnet 34+1=35

Von dieser Endsumme zieht der erste Schüler wieder seine gedachte Zahl ab 35-15=20 und erhält somit die Summe aller Zensuren, die dann nur noch durch die bekannte[xiv] Anzahl der Schüler geteilt werden muss ➔ 20:10=2. Das Ergebnis ist also ein Durchschnitt von einer glatten 2. Keiner der Schüler hat anhand der gehörten Zwischenwerte irgendetwas über die Zensuren der Klassenkameraden erfahren.

Dies ist ein erstes einfaches Beispiel für das Rechnen mit (verschlüsselten[xv]) Werten. Man muss sich jedoch darauf verlassen, dass jeder korrekt rechnet (ohne dies prüfen zu können) und man muss sich auch darauf verlassen, dass sich die Schüler anschließend nicht (heimlich) die genannten Zwischenwerte mitteilen. Wenn z. B. der 3. Schüler die von ihm errechnete 22 dem 5. Schüler anschließend mitteilt, dann kann dieser die Zensur des 4. Schülers (eine 4) bestimmen.

Genau genommen ist dies nicht nur ein Beispiel für homomorphe Verschlüsselung, sondern auch ein Beispiel für „Mehrparteienrechnung“ bzw. „multiparty computing“ (MPC), was aber von Prof. Kelber in dem eingangs genannten „Abschiedsbrief“ auch als spannende Technologie genannt wurde, die man bei den PET im Auge behalten sollte.

2.2 Eine etwas homomorphe Verschlüsselung

Eine Verschlüsselung, bei der ausschließlich eine Rechenart[xvi] mit den verschlüsselten Daten funktioniert, wie es Rivest bei dem RSA-Verfahren bezüglich der Multiplikation entdeckt hatte, reicht für praktische Anwendungen im Normalfall nicht aus. Von einer „etwas homomorphen Verschlüsselung“ (a „somewhat homomorphic encryption“ – SWHE) ist die Rede, wenn man neben der Multiplikation auch noch die Addition von verschlüsselten Daten durchführen kann, so dass das entschlüsselte Ergebnis, dem der Addition bzw. Multiplikation der unverschlüsselten Werte entspricht.

Ähnlich wie bei dem obigen Beispiel mit dem Durchschnitt der Klassenarbeit besteht die Verschlüsselung z. B. darin, dass man den Schlüssel (P)[xvii] zu den Werten (m_i)[xviii] addiert. Die derart verschlüsselten Werte werden entsprechend als „c_i“[xix] bezeichnet. Der Schlüssel P sollte eine große (zufällig gewählte) Zahl sein, die deutlich größer sein muss als alle Klartextwerte – einschließlich der Rechenergebnisse. In diesem Fall ist die Entschlüsselung sehr einfach: Man teilt die Verschlüsselten Werte durch P und erhält einen Rest. Dieser Rest entspricht dem Klartext[xx]. Es gibt dabei nur ein Problem: Auch wenn man eine große Zahl für P wählt, ist diese leicht aus den verschlüsselten c_i bestimmbar, weil alle verschlüsselten Werte c_i um P herum liegen. Um dieses Problem zu lösen, wird P noch mit individuellen (großen, ganzzahligen) Werten n_i multipliziert. Die Berechnung der c_i wäre dann:

$c_i=n_i*P+m_i$.

Die einzelnen Werte von n_i muss man sich nicht merken. Der Ver- und Entschlüsselungsschlüssel ist ausschließlich „P“. Die Entschlüsselung funktioniert trotzdem noch so wie oben

beschrieben. Der Unterschied ist jetzt aber, dass man aus den derart verschlüsselten c_i nicht mehr auf den Schlüssel P schließen kann. Eine verschlüsselte „1“ wäre nicht mehr in der Nähe einer verschlüsselten „2“.

Die folgenden Formeln sollen zeigen, dass die derart verschlüsselten Werte von c_i tatsächlich homomorph bezüglich Addition und Multiplikation sind.

Addition

$c_r=c_1+c_2 = n_1*P+m_1 + n_2*P+m_2$, „$c_r$“ ist das verschlüsselte Ergebnis (result)

$c_r=P*(n_1+n_2) + m_1+m_2$

Wenn man dieses verschlüsselte Ergebnis c_r durch P teilt, dann kommt n_1+n_2 heraus und es bleibt ein Rest von m_1+m_2. Wie oben beschrieben interessiert hier nur der Rest – also m_1+m_2 – was zu beweisen war (wird in der Mathematik auch gern mit „qed“ abgekürzt: „quod erat demonstrandum“).

Die Entschlüsselung, bei der eine ganze Zahl durch eine natürliche Zahl dividiert wird und nur der ganzzahlige Rest als Ergebnis übrigbleibt, wird auch als „Modulo-Funktion“ bezeichnet und mit „mod“ abgekürzt. Die Entschlüsselung entspricht somit:

$c_r \bmod P = m_1+m_2$.

Multiplikation

$c_r=c_1*c_2 = (n_1*P + m_1)*(n_2*P + m_2)$

$c_r=P^2*n_1*n_2 + P*m_2*n_1 + P*m_1*n_2 + m_1*m_2$

$c_r=P*(P*n_1*n_2 + m_2*n_1 + m_1*n_2) + m_1*m_2$

Wenn man c_r entschlüsselt – also durch P dividiert und nur den Rest behält, kommt m_1*m_2 heraus (qed)!

Wie schon erwähnt, funktioniert das nur mit großen Werten von P, die größer sind als alle Werte (einschließlich der Rechenergebnisse) im Klartext. Wie man außerdem sieht, werden die c_i mit jeder Rechnung größer. Speziell nach einer Multiplikation werden die Zahlen, mit denen man rechnen muss, doppelt so lang. Dieser Overhead, der in diesem Zusammenhang auch als „Rauschen“ bezeichnet wird, macht ein Weiterrechnen nach mehreren Operationen unmöglich. Bei der „etwas homomorphen Verschlüsselung“ beschränkt man sich daher auf z. B. fünf Additionen und eine Multiplikation[xxi]. Danach muss jeder c_r-Wert „aufgefrischt“, also wieder entschlüsselt und neu verschlüsselt werden. Wenn man das den Cloud-Anbieter erledigen lassen würde, bräuchte er den Schlüssel – das will man nicht. Wenn man das als Cloud-Nutzer (Verantwortlicher) selbst durchführen müsste, dann wäre dieses permanente Auffrischen aufwändiger, als gleich alles selbst zu rechnen – das will man auch nicht.

Die Lösung der beiden Probleme der SWHE – die Erweiterung der homomorphen Eigenschaften auf beliebige Operationen und die häufig erforderliche Auffrischung der Zwischenergebnisse ohne Zusatzaufwand für den Verantwortlichen und ohne, dass der Cloud-Anbieter den Schlüssel

(im Klartext) kennen muss – kam im Jahr 2009 in Form einer Doktorarbeit von Craig Gentry[xxii].

3. Voll homomorphe Verschlüsselung – fully homomorphic encryption (FHE)

Sowohl die angegebene Doktorarbeit (etwa 200 Seiten) als auch die unter dem Link mit zur Verfügung gestellte Zusammenfassung (8 anspruchsvolle kleingedruckte Seiten) können hier nicht im Detail wiedergegeben werden. Die Grundideen zur Lösung der beiden oben genannten Probleme der SWHE, die zur FHE führen, können aber hier beschrieben werden:

3.1 Die Erweiterung der SWHE zu beliebigen Berechnungen

Wenn man die Addition zweier Bit betrachtet, entspricht die letzte Stelle des Ergebnisses einer XOR-Funktion und die nächste Stelle (der Übertrag) einer UND-Funktion: 0+0=0, 0+1=1, 1+0=1, 1+1=10 (im Dualsystem).[xxiii] Die Multiplikation ist noch einfacher. Sie kann mit einer einzigen UND-Funktion realisiert werden: 0*0=0, 0*1=0, 1*0=0, 1*1=1. Das ist bereits das gesamte Einmaleins im Dualsystem. Weil es so einfach ist, rechnen Computer im Dualsystem.

Man kann das Pferd aber auch von hinten aufzäumen und sagen: Wenn man die Addition und die Multiplikation beherrscht (und das ist mit SWHE bereits der Fall), dann kann man damit auch die XOR- und die UND-Funktionen realisieren und damit dann auch die ODER- und die Nicht-Funktion. Alles, was ein Computer berechnet, wird letztlich durch viele aufeinanderfolgende dieser „logischen“ Grundoperationen und deren Kombinationen durchgeführt.[xxiv]

Mit den homomorph verschlüsselten Bits lassen sich also alle logischen Schaltungen eines Computers berechnen, was letztlich bedeutet, dass alle Operationen eines Computers homomorph durchgeführt werden können. Dass dies einen enormen Mehraufwand gegenüber den ursprünglichen UND-, ODER und NICHT-Funktionen bedeutet, steht außer Frage, aber dazu später mehr. Zunächst steht fest: Es geht.

3.2 Auffrischung der Zwischenergebnisse

Das zweite zu lösende Problem des SWHE war die regelmäßige „Auffrischung“ der verschlüsselten Zwischenergebnisse, um deren „Rauschen“ zu reduzieren.

Dabei ging es darum, dass man einerseits diese Aufgabe als Auftraggeber nicht selbst erledigen möchte, weil man dann mehr Aufwand mit dem Auffrischen hätte, als man durch den Auftragnehmer (heute wohl ein Cloud-Anbieter) erledigen lassen könnte[xxv] und dass man andererseits dem Auftragnehmer den Schlüssel nicht (im Klartext) zur Verfügung stellen möchte – weil ansonsten die gesamte homomorphe Verschlüsselung keinen Sinn mehr machen würde.

Der Lösungsansatz besteht nun darin, dass der Schlüssel selbst noch einmal homomorph verschlüsselt wird. Der Auftragnehmer sieht ihn also nicht (im Klartext), kann damit aber die Auffrischung durchführen. Bei dieser Auffrischung bleiben die eigentlichen Werte zu jedem Zeitpunkt für den Auftragnehmer verschlüsselt.

Das ist in wenigen Worten ausgedrückt die Lösung von Gentry in dessen Doktorarbeit aus dem Jahr 2009. Die Mathematik dahinter sprengt jedoch die Möglichkeiten, dies in einem kurzen, für jeden Datenschützer verständlichen Artikel auszuführen. Es gibt aber ein anschauliches Beispiel, das Gentry bereits in seiner Doktorarbeit (und auch in der unter dem angegebenen Link mit enthaltenen 8-Seitigen Zusammenfassung) erläutert hat: *„Alice’s Jewelry Store“*. Bei dieser

Analogie spielt wieder eine Kiste eine wichtige Rolle, so wie auch schon bei der Veranschaulichung des RSA-Verfahrens.

Die Geschichte zur Lösung des Problems der Auffrischung der verschlüsselten Werte als Analogie zu „Alice' Juweliergeschäft“ ist folgendermaßen:

Alice hat ein Juweliergeschäft und einen Angestellten, der für sie Schmuckstücke z. B. aus einem Klumpen Gold anfertigen soll. Dieser Angestellte ist handwerklich sehr gut, so dass sich Alice sicher sein kann, dass er aus dem Gold z. B. einen schönen Ring anfertigen kann. Alice hat aber kein so unbegrenztes Vertrauen in den Angestellten, dass sie ihm das Gold direkt (ungeschützt) geben will (das kommt uns bekannt vor – oder?). Sie will, dass der Angestellte das Gold **verarbeiten** kann, **ohne direkten Zugriff** darauf zu haben (das kommt uns auch bekannt vor).

Ihr Lösungsansatz: Sie legt den Klumpen Gold in eine abschließbare Kiste, deren Schlüssel nur sie selbst hat. Die Kiste enthält aber zwei Löcher mit Handschuhen[xxvi], so dass der Angestellte seine Hände in die Handschuhe stecken und somit das Gold mit den Händen in den Handschuhen bearbeiten kann. Herausholen kann er aber weder das Gold noch Teile davon und auch nicht den von ihm selbst hergestellten Ring. Es gibt mit dieser Kiste bzw. mit den Handschuhen noch einen Haken: Die Handschuhe werden durch die Bearbeitung immer steifer, so dass der Angestellte damit nur eine beschränkte Anzahl vor Arbeitsschritten durchführen kann, die aber für eine Herstellung des Rings nicht ausreichen (das kommt uns schon wieder bekannt vor).

Die Lösung von Alice für dieses Problem: Sie hat eine weitere Handschuhkiste, die ähnlich ist, wie die erste, nur etwas größer und mit einem zusätzlichen Schlitz (ähnlich wie an einem Briefkasten), durch den die erste Kiste in die Zweite Kiste gesteckt (aber nicht wieder herausgeholt) werden kann. In diese zweite Kiste hat Alice den Schlüssel für die erste Kiste gelegt und übergibt sie ihrem Angestellten. Der Angestellte nimmt jetzt die erste Kiste und steckt sie durch den Schlitz in die zweite. Der Schlüssel für die erste Kiste kann jetzt vom Angestellten genutzt, aber nicht aus der zweiten Kiste herausgeholt werden, weil dieser Schlüssel ja mit dem Schlüssel der zweiten (verschlossenen) Kiste „verschlüsselt“ ist. Der Angestellte kann jetzt die erste Kiste (in der zweiten Kiste) mit dem dazugehörigen Schlüssel öffnen, das Gold und den Ring herausholen und über die neuen – jetzt wieder geschmeidigen – Handschuhe der zweiten Kiste seine Arbeit fortsetzen, bis auch diese Handschuhe wieder steif werden. Dieser Vorgang kann mit einer dritten, vierten u.s.w. Kiste beliebig lange fortgesetzt werden, bis der Ring fertig ist. Danach übergibt der Angestellte die letzte Kiste an Alice, die dann diese Kiste mit dem nur ihr zur Verfügung stehenden Schlüssel öffnen kann. Dann holt sie das Ergebnis (den fertigen Ring) aus der Kiste.

Die Analogie des Problems des „Auffrischens“ der homomorph verschlüsselten Zwischenergebnisse mit den immer steifer werdenden Handschuhen veranschaulicht den Lösungsansatz von Gentry, ohne dabei zu tief in die dafür erforderliche Mathematik einzusteigen.

Das Problem der FHE ist damit seit 2009 „grundsätzlich“ gelöst. Es ist, wie man sich denken kann, aber nur eine theoretisch nutzbare Lösung, denn praktisch ist das beschriebene Verfahren so aufwändig (ursprünglich in einer Größenordnung um mehr als den Faktor 1000 aufwändiger als die Verarbeitung ohne homomorphe Verschlüsselung), dass sich unter diesen Umständen ein Cloud-Einsatz nicht lohnen würde. Die Cloud-Lösungen sind ja deshalb so beliebt, weil sie billiger sind als der Betrieb eigener Rechenzentren. Sie sind aber nicht um einen Faktor >1000 billiger.

Weiterentwicklungen - Ausblick

Die Tatsache, dass die von Gentry ursprünglich vorgeschlagene Lösung keine praktische Relevanz hat, bedeutet nicht, dass die Überlegungen zur homomorphen Verschlüsselung damit zu Ende gingen oder in eine Sackgasse führten. Ganz im Gegenteil! Nachdem man erst einmal wusste, dass es prinzipiell funktioniert, ging es „nur“ noch darum das vorgestellte Verfahren zu optimieren. Die Literatur zur homomorphen Verschlüsselung, die nach 2009 veröffentlicht wurde, bezieht sich grundsätzlich auf die Doktorarbeit von Gentry, weil dies der Durchbruch zur Umsetzung der Idee von Rivest zu einer FHE war.

In den anschließenden Jahren gab es viele neue Arbeiten mit Optimierungen des von Gentry vorgestellten Ansatzes. Hier ist eine knappe Aufzählung, ohne weitere Erläuterungen:

- FHEW – Fastest Homomorphic Encryption in the West
- FFHE – Faster Homomorphic Encryption
- TFHE – Fully Homomorphic Encryption over the Torus

Diese und viele weitere Optimierungen seit 2009 haben zu einer beträchtlichen Aufwandsreduzierung geführt. Belastbare Werte sind dem Autor nicht bekannt, aber man kommt offensichtlich in den Bereich, ab dem sich der praktische Einsatz zu lohnen beginnt. In einem Buch über moderne Kryptographie[xxvii] aus dem Jahr 2024 steht in der Zusammenfassung von Kap. 11 „Homomorphic Encryption“ folgendes:

„Homomorphic encryption has gained considerable momentum over the past 10 years. The need to secure data from third-party vendors, CSPs[xxviii], and auditors while allowing them to process it simultaneously is rapidly increasing. Homomorphic encryption is one of the main solutions in sight right now. The increased use of cloud computing will fuel the need for homomorphic encryption and its development.“

Übersetzung:

„Die homomorphe Verschlüsselung hat in den letzten 10 Jahren erheblich an Bedeutung gewonnen. Der Bedarf, Daten von Drittanbietern, CSPs und Prüfern zu schützen und ihnen gleichzeitig die Verarbeitung dieser Daten zu ermöglichen, nimmt rapide zu. Die homomorphe Verschlüsselung ist derzeit eine der wichtigsten Lösungen. Die zunehmende Nutzung von Cloud Computing wird den Bedarf an homomorpher Verschlüsselung und deren Entwicklung weiter ankurbeln.“

Abgesehen von der (anspruchsvollen) Verwirklichung einer FHE gibt es auch vielversprechende Ansätze für konkrete Einsatzgebiete von HE, die dann nicht „voll“ (universell), sondern nur für spezielle Probleme einsetzbar sind. Die Lösungen sind dann von deutlich geringerem Aufwand. Die Grundgedanken der HE bleiben dabei aber erhalten. In diesem Zusammenhang sollen nur kurz zwei bereits von Microsoft und Apple umgesetzte Anwendungsfälle genannt werden:

- Microsoft Edge – Abgleich von (homomorph verschlüsselten) Anmeldedaten mit öffentlichen Leak-Datenbanken. Microsoft kann dabei die Anmeldedaten nicht im Klartext lesen.
- Apple Bild-Attributierung – dabei werden nicht die Bilder selbst, sondern nur homomorph verschlüsselte Attribute vom iPhone an Apple gesendet. Näheres dazu gibt es in einem Podcast[xxix] von Heise, sowie in einem dort genannten weiteren Artikel[xxx], der auch auf einen Beitrag von Apple[xxxi] selbst referenziert.

Dieser Artikel des AK-Krypto des BvD soll nur einen Einstieg in die homomorphe Verschlüsselung für Interessierte geben. Für einen tieferen Einblick in die Thematik wird auf die angegebenen Referenzen verwiesen. Wer konkret mit homomorpher Verschlüsselung arbeiten möchte, der kann sich unter den zahlreichen open-source Bibliotheken[xxxii] umsehen, in denen FHE bereits implementiert ist.

---

[i] „Digital, frei und gleich – Um was es geht“ https://www.bfdi.bund.de/SharedDocs/Downloads/DE/DokumenteBfDI/Reden_Gasteitr%C3%A4ge/2024/Abschiedsbrief-Kelber.pdf?__blob=publicationFile&v=1

[ii] Insbesondere die Mathematiker und Informatiker unter den Datenschützern, ggf. aber sogar bis hin zu den Juristen.

[iii] Ursprünglich waren alle Verschlüsselungsverfahren „symmetrisch“, so dass dies in der Vergangenheit nicht explizit dazu gesagt werden musste. Gemeint sind alle Verfahren, bei denen für die Ver- und die Entschlüsselung der gleiche Schlüssel genutzt wird.

[iv] https://de.wikipedia.org/wiki/Advanced_Encryption_Standard

[v] ISBN 9-783423-330718

[vi] Das Verfahren ist nach seinen „Entdeckern“ benannt: Ronald L. **R**ivest, Adi **S**hamir und Leonard **A**dleman

[vii] FISA: https://de.wikipedia.org/wiki/Foreign_Intelligence_Surveillance_Act
Cloud Act: https://de.wikipedia.org/wiki/CLOUD_Act

[viii] https://de.wikipedia.org/wiki/Datenschutzrahmen_EU-USA

[ix] Falls es doch Software des Cloud-Anbieters ist, sollte wenigstens der Teil der Verschlüsselung Open Source und somit für jedermann überprüfbar sein.

[x] Von altgriechisch ὁμός homós „gleich“ und μορφή morphé „Form, Gestalt“ – wird in der Mathematik auch im Sinne von „strukturerhaltend“ verwendet: https://de.wikipedia.org/wiki/Homomorphismus
bzw. „Man sagt auch, dass ein Homomorphismus eine *verknüpfungstreue* Abbildung ist.“ – Quelle: „Basiswissen Zahlentheorie: Eine Einführung in Zahlen und Zahlbereiche“, Kap. 6.3, S. 148, ISBN 978-3-642-39773-8

[xi] https://luca-giuzzi.unibs.it/corsi/Support/papers-cryptography/RAD78.pdf

[xii] „Datenschutz geht zur Schule“ (DSgzS) ist eine Initiative des BvD, über die Datenschützer ehrenamtlich Datenschutzvorträge in Schulen halten, um die Schüler bezüglich möglicher Gefahren bei unbedachter Verbreitung der eigenen Daten z. B. in sozialen Medien frühzeitig zu informieren/sensibilisieren. URL: www.dsgzs.de

[xiii] In der Praxis würde man sich wohl auf eine natürliche Zahl beschränken aber theoretisch funktioniert es auch mit negativen ganzen Zahlen.

[xiv] Die Schüler sind schließlich im Kreis aufgestellt und können leicht abgezählt werden. Falls Schüler fehlen, die an der Deutscharbeit teilgenommen hatten, bezieht sich das Ergebnis selbstverständlich nur auf den Durchschnitt der anwesenden Schüler.

[xv] Der Schlüssel wäre in diesem Fall die gedachte 15 des ersten Schülers. Er hat aber letztlich genauso wenig Information über die Zensuren der Klassenkameraden wie diese.
[xvi] Das wird in der Literatur als „Partial Homomorphic Encryption“ (PHE) bezeichnet.
[xvii] Man benutzt hier teilweise Primzahlen, aber bei Gentry wird nur eine ungerade natürliche Zahl verlangt, worauf hier aber nicht näher eingegangen werden soll.
[xviii] Die unverschlüsselten Werte (Message) werden als „m“ bezeichnet. Da es viele zu verschlüsselnde Werte gibt, werden diese mit „m_i“ bezeichnet.
[xix] Das „c“ steht für „cipher“. Für jeden Klartextwert m_i gibt es einen verschlüsselten Ciphertext c_i.
[xx] Bei diesem einfachen Beispiel könnte man P auch direkt vom verschlüsselten Wert wieder abziehen und würde als Differenz den Wert im Klartext erhalten. Wir bleiben aber bei der ganzzahligen Division, bei der uns nur der Rest interessiert, weil das auch bei den Beispielen in den folgenden Abschnitten noch funktioniert.
[xxi] Es gibt unterschiedliche Varianten.
[xxii] „A FULLY HOMOMORPHIC ENCRYPTION SCHEME“ https://crypto.stanford.edu/craig/
[xxiii] XOR steht dabei für das „exklusive oder“: 0 XOR 0=0, 0 XOR 1=1, 1 XOR 0=1, 1 XOR 1=0. Das „exklusive oder“ hat somit die Bedeutung für „das eine **oder** das andere, aber **nicht beides**.
Die UND-Funktion steht für „das eine **und** das andere“: 0 AND 0 = 0, 0 AND 1 = 0, 1 AND 0 = 0, 1 AND 1 =1.
Die (einfache) ODER-Funktion steht für „das eine **oder** das andere“: 0 OR 0 = 0, 0 OR 1 = 1, 1 OR 0 = 1, 1 OR 1 = 1. Gentry hat hier zusätzlich noch mod 2 gerechnet, so dass es keinen Übertrag gibt. Die Addition zweier Bits entspricht dann tatsächlich einem XOR!
[xxiv] Kombinationen sind z. B. NAND (NOT in Verbindung mit AND) oder NOR (NOT in Verbindung mit OR).
Die NOT-Funktion steht für das Gegenteil: NOT 0 = 1, NOT 1 = 0 – damit erschließt sich auch der Titel eines Films über die Anfänge des CCC: *„Alles ist Eins. Ausser der 0.“*
NAND steht für AND mit anschließendem NOT: 0 NAND 0 = 1, 0 NAND 1 = 1, 1 NAND 0 = 1, 1 NAND 1 = 0
NOR steht für OR mit anschließendem NOT: 0 NOR 0 = 1, 0 NOR 1 = 0, 1 NOR 0 = 0, 1 NOR 1 = 0.
Genau genommen braucht man sogar nur eine einzige dieser „logischen“ Funktionen (NAND oder NOR), um damit auch alle anderen zu generieren.
[xxv] Wenn man es selbst machen würde, bräuchte man den gesamten Aufwand mit der homomorphen Verschlüsselung nicht, denn man selbst hätte die Daten ja im Klartext.
[xxvi] Siehe z. B. https://de.wikipedia.org/wiki/Handschuhkasten
[xxvii] Modern Cryptography – The Practical Guide, ISBN 978-1-4932-2565-1
[xxviii] **C**loud **S**ervice **P**rovider
[xxix] https://www.heise.de/news/Passwort-Folge-26-Homomorphe-Verschluesselung-10282388.html
[xxx] Dieser Beitrag liegt hinter der Heise-Bezahlschranke – er enthält das, was auch im Podcast beschrieben wird: https://www.heise.de/hintergrund/Wie-Apple-mit-homomorpher-Verschluesselung-Daten-schuetzen-will-10193623.html
[xxxi] https://machinelearning.apple.com/research/homomorphic-encryption
[xxxii] FHE Libraries: https://github.com/jonaschn/awesome-he