Informatik Spektrum (2022) 45:137–145
https://doi.org/10.1007/s00287-022-01440-9

HAUPTBEITRAG

Privatsphärefreundliches maschinelles Lernen

Teil 2: Privatsphäreangriffe und Privacy-Preserving Machine Learning

Joshua Stock[1] · Tom Petersen[1] · Christian-Alexander Behrendt[2] · Hannes Federrath[1] · Thea Kreutzburg[2]

Angenommen: 8. Februar 2022 / Online publiziert: 10. März 2022

Zusammenfassung

Verfahren des maschinellen Lernens (ML) beruhen auf dem Prinzip, dass ein Algorithmus Muster und statistische Zusammenhänge in Datensätzen erkennt, diese in einem Modell abbildet und das Modell anschließend auf andere Datensätze anwenden kann. Neben den großen Chancen, die maschinelle Lernverfahren mit sich bringen, birgt diese Technologie allerdings auch Risiken für die Privatsphäre, die in diesem Artikel in Form von Privatsphäreangriffen beleuchtet werden.
Angriffe wie *Model Inversion* zielen auf oftmals sensible Informationen ab, die sich während der Trainingsphase eines ML-Algorithmus ungewollt in einem Modell etabliert haben. Wenn Trainingsdaten Personenbezug aufweisen, insbesondere wenn es sich etwa um vertrauliche medizinische Daten handelt, kann dies problematisch für Betroffene sein.
Demgegenüber stehen Techniken des privatsphärefreundlichen maschinellen Lernens wie *Federated Learning*, die eine Risikominimierung für ein breites Spektrum an Privatsphäreverletzungen ermöglichen. Ausgewählte Techniken aus diesem Bereich werden in diesem Artikel ausführlich dargestellt.
Dies ist der zweite Teil einer zweiteiligen Artikelserie, deren Auftakt unter dem Titel *Grundlagen und Verfahren* bereits in der letzten Ausgabe des Informatik Spektrums erschienen ist.

Einleitung

Mit dem vermehrten Einsatz von Verfahren des maschinellen Lernens (ML) in zahlreichen Bereichen der Gesellschaft werden zunehmend auch Kontexte erschlossen, in denen personenbezogene Daten verarbeitet werden. Darunter sind auch immer häufiger sensible Daten wie etwa Gesundheits- oder Sozialdaten.

✉ Joshua Stock
joshua.stock@uni-hamburg.de

Tom Petersen
tom.petersen@uni-hamburg.de

Christian-Alexander Behrendt
ch.behrendt@uke.de

Hannes Federrath
hannes.federrath@uni-hamburg.de

Thea Kreutzburg
t.kreutzburg@uke.de

[1] Universität Hamburg, Hamburg, Deutschland

[2] Universitätsklinikum Hamburg-Eppendorf, Hamburg, Deutschland

Dabei kann es beim Einsatz von ML aus Sicht der Betroffenen zu verschiedenen Gefährdungen kommen, die der Verarbeitung von personenbezogenen Daten teilweise innewohnen, teils aber auch erst durch den Einsatz von ML entstehen. In diesem Artikel werden primär Gefahren für das Schutzziel Vertraulichkeit, also den Schutz vor unberechtigter Kenntnisnahme von Daten, adressiert. Werden z. B. besonders sensible personenbezogene Daten für das Training der Verfahren verwendet, so kann ein unberechtigter Zugriff auf diese Daten negative Konsequenzen für Betroffene mit sich bringen. Aber auch eine Verletzung der Vertraulichkeit von Inferenzergebnissen, etwa der unberechtigte Zugriff auf ein durch ML-Verfahren vorhergesagtes Risiko für eine spezifische Erkrankung, kann Betroffenen schaden.

Einen Überblick über geeignete Maßnahmen zur Minimierung dieser Risiken, zusammengefasst unter dem Begriff *Privacy-Preserving Machine Learning*, bietet der letzte Abschnitt dieses Artikels. Zunächst jedoch werden im folgenden Abschnitt Gefahren für die Privatsphäre von Betroffenen, deren Daten für das Training von ML-Modellen verwendet wurden, geschildert. ML birgt Risiken für die Privatsphäre. Eine Teilmenge dieser Risiken besteht aus Pri-

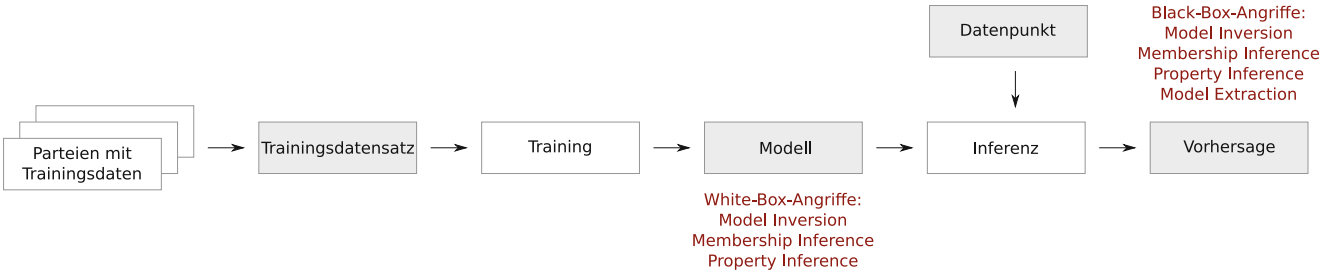

Abb. 1 Eine schematische ML-Pipeline mit möglichen Angriffspunkten für Privatsphäreangriffe

vatsphäreangriffen. Diese Teilmenge wird in diesem Artikel diskutiert.

Angriffe auf maschinelle Lernverfahren

Den betrachteten Angriffen auf die Privatsphäre liegt meist folgendes Szenario zugrunde: Angreifende haben Zugriff auf ein bereits trainiertes ML-Modell und versuchen, hieraus unrechtmäßig Informationen zu extrahieren, die über die gewöhnliche Anwendung des Modells hinausgehen. Damit kann die Privatsphäre von Betroffenen, deren Daten für das Training des Modells verwendet wurden, gefährdet sein. Dies gilt insbesondere, wenn ein Modell im Anschluss an die Trainingsphase von anderen Parteien uneingeschränkt genutzt werden kann.

Prinzipiell sind Privatsphäreangriffe auf verschiedene ML-Algorithmen möglich. Im Fokus der aktuellen Forschung stehen vor allem neuronale Netze, wenngleich die meisten Angriffe auch für andere Verfahren realisiert wurden. Eine Übersicht über bisher veröffentlichte Arbeiten zu Privatsphäreangriffen auf verschiedene ML-Verfahren bieten Rigaki und Garcia [33].

Unabhängig von den ML-Verfahren wird grundsätzlich zwischen zwei Angriffsszenarien unterschieden: *White-Box* und *Black-Box*. Im Black-Box-Szenario (siehe Abb. 1) können Angreifende ein ML-Modell nutzen, um Inferenzen (z. B. Vorhersagen oder Klassifizierungen) durchzuführen. Wie bei einem Orakel können dabei Datensätze an das Modell geschickt werden, um die Modellausgaben zu erhalten. Dabei steht aber lediglich ein Interface zur Verfügung, hinter dem die Modellinterna, also die Hyperparameter (siehe Teil 1 im vorhergehenden Heft) und die antrainierten Parameter, versteckt sind. Im Gegensatz dazu bedeutet White-Box-Zugriff komplette Einsicht in das ML-Modell: Neben der Möglichkeit, mit dem Modell Inferenzen durchzuführen, haben Angreifende hiermit vollen Zugriff auf die Modellparameter und -hyperparameter. Als Abstufung zwischen diesen beiden Extremen sind verschiedene *Gray-Box*-Szenarien möglich. Hierzu könnte beispielsweise die Einsicht in manche Hyperparameter oder die Architektur eines neuronalen Netzes zählen, ohne Zugriff auf andere Parameter.

Wie ebenfalls aus der Übersicht von Rigaki und Garcia [33] hervorgeht, sind die meisten ML-Privatsphäreangriffe per Black-Box-Zugriff realisierbar und benötigen daher keinen Vollzugriff auf das Modell. Dieses Szenario ist insbesondere im Kontext von *Machine Learning as a Service* (MLaaS) oftmals gegeben. MLaaS ist ein Sammelbegriff für Dienstleistungen, die bei der Entwicklung von ML-Modellen unterstützen. Angefangen bei der Installation von Software, über das Preprocessing von Daten bis hin zur Trainingsphase selbst können damit ressourcenintensive Schritte zu kommerziellen Dienstleistern ausgelagert werden. MLaaS-Provider können ihren Kundinnen und Kunden also neben Hardware, vorinstallierten Arbeitsumgebungen und Algorithmen auch fertige Modelle zur Verfügung stellen. Neben spezialisierten ML-Plattformen (z. B. *BigML*) bieten mittlerweile auch alle großen Cloud-Computing-Anbieter wie *Google Cloud* MLaaS an. Entsprechende Privatsphäreangriffe können folglich auch auf ML-Modelle, die zu diesen Anbietern ausgelagert wurden, gerichtet sein.

Die folgenden Abschnitte widmen sich detailliert den einzelnen ML-spezifischen Angriffen, die in Abb. 1 in einer Übersicht zusammengefasst sind.

Model Inversion

Model Inversion bietet Angreifenden die Möglichkeit, durch Anfragen an ein Modell Teile der Trainingsdaten zu rekonstruieren, ohne dabei tiefgreifendes Vorwissen über die Daten zu benötigen. Dies geschieht in der Regel per Black-Box-Zugriff, erweiterte Angriffszenarien in White- bzw. Gray-Box-Szenarien sind jedoch ebenfalls möglich [37]. Ein bekanntes Beispiel ist in Abb. 2 zu sehen. Fredrikson u. a. [11] gelang hier die Rekonstruktion eines Portraitfotos mit erkennbaren Merkmalen aus den Trainingsdaten eines Gesichtserkennungsmodells [11]. In bestimmten Szenarien genügt es demnach, wiederholt gezielte Anfragen an ein Modell zu stellen, um Informationen aus den Trainingsdaten zu rekonstruieren. Diese Art des Angriffs wird seit der Veröffentlichung weiter erforscht und es wurden bereits verschiedene Varianten und Weiter-

Abb. 2 Ein Beispiel des *Model-Inversion*-Angriffs mit einer Rekonstruktion (*links*) des Originalbilds (*rechts*) aus den Trainingsdaten. Abbildung entnommen aus [11]

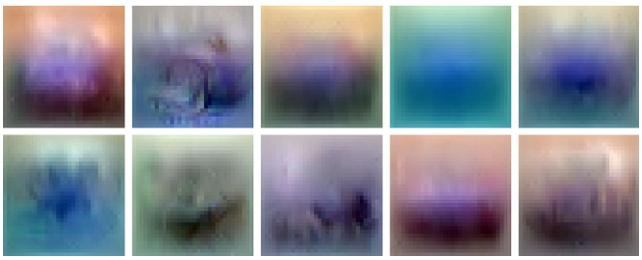

Abb. 3 Die Ergebnisse von Shokri u. a. [36], die den *Model-Inversion*-Angriff auf ein Bilderkennungsmodell für den Datensatz CIFAR-10 ausgeführt haben. Hier sind Rückschlüsse auf die Trainingsdaten kaum möglich: Die Darstellungen zeigen die rekonstruierten Daten für die jeweiligen Zielklassen des angegriffenen Modells (*von oben links:* Flugzeug, Auto, Vogel usw.). Abbildung entnommen aus [36]

entwicklungen publiziert (z. B. [40]). Allerdings scheinen erfolgreiche Model-Inversion-Angriffe nur in bestimmten Szenarien möglich zu sein: So zeigen etwa Shokri u. a. [36], dass der Angriff lediglich den Durchschnitt der Trainingsdaten einer Datenklasse aufdecken kann, was in vielen Anwendungsfällen zu unbrauchbaren Ergebnissen führt [36]. Dieser Zusammenhang wird durch Abb. 3 verdeutlicht.

Membership Inference

Das Ziel eines *Membership-Inference*-Angriffs [36] ist es, für einzelne Datenpunkte zu entscheiden, ob sie Teil des Trainingsdatensatzes eines ML-Modells waren. Das Anwendungsgebiet eines ML-Algorithmus ist hierbei ausschlaggebend dafür, inwieweit der Angriff zu Privatsphäreverletzungen führen kann. Betrachtet man etwa einen Algorithmus, der Personen, die an einer seltenen genetischen Grunderkrankung leiden, die Wahrscheinlichkeit für eine notwendige Operation vorhersagt, dann kann in diesem Szenario bereits das Wissen über die bloße Zugehörigkeit zum Trainingsdatensatz privatsphäreverletzend sein.

Auch dieser Angriff kann per Black-Box-Zugriff realisiert werden: Zugrunde liegt hierbei die Idee, dass sich die Ausgaben des Modells für arbiträre Daten etwas von Ausgaben für jene Daten unterscheiden, mit denen das Modell trainiert wurde. Da das Modell während des Trainings schrittweise an die Trainingsdaten angepasst wurde, sind Vorhersagen für Trainingsdaten in der Regel exakter bzw. eindeutiger.

Eine zunächst wirksame Methode gegen Membership-Inference-Angriffe ist es, die Ausgabe von ML-Modellen zu reduzieren, sodass in einer Klassifizerungsaufgabe statt Wahrscheinlichkeitswerten für Klassenzugehörigkeiten lediglich der Name der wahrscheinlichsten Klasse ausgegeben wird („label-only“). Jedoch sind Membership-Inference-Angriffe gut erforscht und es wurden Varianten entwickelt, die auch für Label-only-ML-Algorithmen funktionieren [7]. Andere Arbeiten erlauben eine Erhöhung der Angriffsgenauigkeit im White-Box-Szenario, indem zusätzlich Informationen aus den Modellparametern berücksichtigt werden [22].

In der Literatur werden die beiden Fälle *Zugehörigkeit* und *Nicht-Zugehörigkeit* eines einzelnen Datenwertes zu den Trainingsdaten in der Regel scharf voneinander abgegrenzt: Angreifende evaluieren ihren Angriff für Daten, die exakt in der vorliegenden Form Teil der Trainingsdaten waren, bzw. für Daten, die sich maßgeblich von Trainingsdaten unterscheiden. Membership Inference wäre aber auch für Daten denkbar, die Trainingsdaten *ähneln* – etwa für Fotos aus einer leicht veränderten Perspektive oder tabellarische Daten mit geringer Unschärfe. Angreifende könnten dann mehrere Anfragen mit jeweils verschiedenen Varianten der Informationen, die über einen potenziellen Trainingsdatenwert vorliegen, an das Modell stellen. Eine systematische Auswertung der Modell-Rückgabewerte könnte dann ebenfalls Rückschlüsse auf die verwendeten Trainingsdaten zulassen.

Property Inference

Ähnlich wie Membership Inference zielen auch *Property-Inference*-Angriffe auf die einem Modell zugrunde liegenden Trainingsdaten ab [2]. Die meisten Property-Inference-Angriffe haben das Ziel, globale Eigenschaften der Trainingsdaten anhand des Modellverhaltens (im Black-Box-Szenario) oder anhand der Modellparameter (im White-Box-Szenario) zu extrahieren. Da diese Eigenschaften oft nicht in einem Zusammenhang zu der vom Modell gelernten Aufgabe stehen, können sie mehr über das Modell bzw. dessen Trainingsdaten verraten als von ihren Modellbesitzern beabsichtigt.

Beispielsweise können per Property Inference stochastische Eigenschaften in den Trainingsdaten wie die Geschlechter- oder Altersverteilung aus Modellen extrahiert

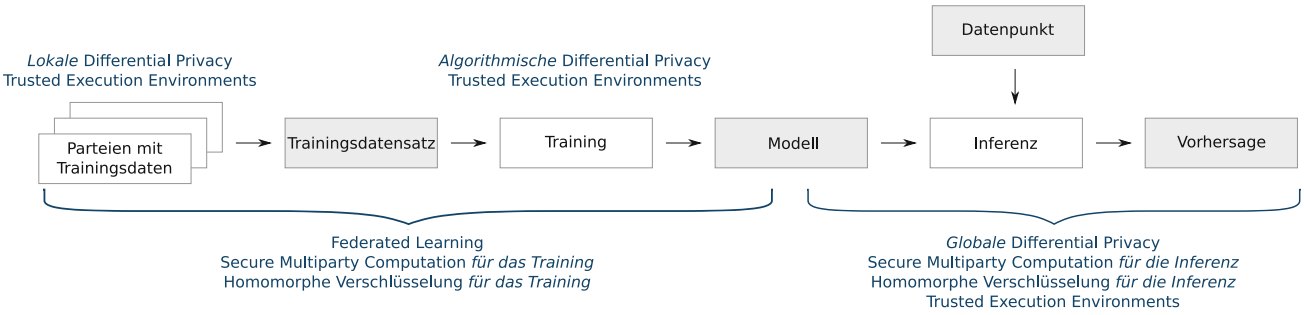

Abb. 4 Verfahren des *Privacy-Preserving Machine Learning* und wo sie zum Einsatz kommen.

werden [12]. In einigen Fällen kann dies problematisch sein, z. B. wenn eine Firma ein ML-Modell mit ihren Kundendaten trainiert und das Modell im Anschluss veröffentlichen möchte, aber keine Rückschlüsse auf die Demografie ihres Kundenstamms möglich sein sollen. Für kollaborative Lernszenarien sind Property-Inference-Angriffe von besonderer Bedeutung, da hier die Privatsphäre einzelner Teilnehmer explizit geschützt werden soll und das Extrahieren von Trainingsdateneigenschaften diese verletzen kann [26].

Model Extraction

Ziel des *Model-Extraction*-Angriffs ist die Übertragung eines Zielmodell-Verhaltens auf ein eigenes Modell, um die aufwendige Trainingsphase für das eigene Modell zu umgehen [31]. Somit ist dieser Angriff ausschließlich für das Black-Box-Szenario konzipiert, da in einem White-Box-Szenario die Modellparameter ohnehin bekannt sind. Insbesondere im MLaaS-Kontext kann dieser Angriff kritisch sein, da hier oft für den Zugriff auf ein Modell bezahlt werden muss und dies durch das bösartige Anfertigen eines gleichwertigen Modells unter Umständen umgangen werden kann. Somit zielt dieser Angriff eher auf Betriebsgeheimnisse ab als auf sensible Informationen, die aus datenschutzrechtlichen Gründen geschützt werden müssen.

Angreifbarkeit von Modellen

Nicht jedes ML-Modell ist anfällig für die genannten Privatsphäreangriffe. Die Angreifbarkeit eines Modells hängt von vielen Faktoren ab: Beispielsweise wird sie beeinflusst von der Struktur und der Größe des Trainingsdatensatzes, der Anzahl der Trainingsiterationen und der dabei verwendeten Lernrate (siehe Teil 1 im vorhergehenden Heft). Abgesehen von der Empfehlung, die Überanpassung von Modellen zu vermeiden, lassen sich kaum pauschale Aussagen zu Privatsphäreschwachstellen treffen. Daher ist es sinnvoll, entwickelte Modelle individuell zu überprüfen. Die frei zugänglichen Werkzeuge *PrivacyRaven* [19] und *ML Privacy Meter* [29] können dabei helfen, indem sie ein gegebenes Modell automatisiert per Membership Inference oder Model Extraction angreifen und die jeweilige Angreifbarkeit quantifizieren. Dies kann in manchen Fällen auch dabei unterstützen, durch Vorher-Nachher-Analysen den Erfolg von Gegenmaßnahmen, wie sie etwa im nächsten Abschnitt beschrieben werden, zu verifizieren.

Privacy-Preserving Machine Learning

Aufgrund der mit dem Einsatz von ML-Verfahren verbundenen Risiken, auch durch die im vorangehenden Abschnitt beschriebenen Privatsphäreangriffe, gibt es zahlreiche Bestrebungen, ML datenschutzgerechter zu gestalten. *Privacy-Preserving Machine Learning* (PPML) ist ein Sammelbegriff für verschiedene Techniken, die zu diesem Ziel beitragen können. Je nach Bedarf können diese Techniken verschiedene Schutzziele in der Trainingsphase und in der Inferenzphase verfolgen. Beispielsweise kann der Einsatz von *Federated Learning* in einem kollaborativen Szenario die zum Training verwendeten Daten vor anderen Parteien schützen. Die Nutzung von *homomorpher Verschlüsselung* in der Inferenzphase ermöglicht die Verwendung eines ML-Verfahrens, ohne dass Nicht-Berechtigte Zugriff auf Eingabe- oder Inferenzdaten erhalten. Abb. 4 bietet einen Überblick darüber, welche Mechanismen für welche Aspekte in ML-Verfahren eingesetzt werden können. Die einzelnen Techniken werden in den folgenden Abschnitten näher vorgestellt, eine Übersicht befindet sich in Tab. 1.

Differential Privacy

Differential Privacy (DP) ist eine von Dwork u. a. [10] entwickelte Metrik [10], die sich in den vergangenen Jahren zum „de-facto Standard für privatsphärefreundliche Datenanalyse“ entwickelt hat [32]. Das Ziel von DP ist es, die Genauigkeit der Datenanalyse zu maximieren, während die Identifizierbarkeit einzelner Individuen in den zugrunde liegenden Daten minimiert wird. Dabei folgt DP der Idee, dass Außenstehende bei einer Auswertung auf einem *differentially private* Datensatz nicht ermitteln können, ob sich die Daten eines bestimmten Individuums im Datensatz befun-

Tab. 1 Übersicht über die vorgestellten Verfahren des Privacy-Preserving-Machine-Learning

Technik	Phase der Anwendung	Funktion	Schwächen
Differential Privacy	Training und/oder Inferenz	Minimiert Einfluss einzelner Datenpunkte	Mindert Transparenz; nicht für alle Datentypen geeignet
Federated Learning	Verteiltes Training	Gemeinsames Training ohne Rohdatenaustausch	Viele Angriffe weiterhin möglich
Secure Multiparty Computation	Verteiltes Training und/oder Inferenz	Gemeinsame Funktionsberechnung ohne Rohdatenaustausch	Häufig rechenintensiv
Homomorphe Verschlüsselung	Training und/oder Inferenz	Berechnungen auf verschlüsselten Daten	Häufig rechenintensiv (FHE); Ersetzung bestimmter Funktionen notwendig
Trusted Execution Environments	Training und/oder Inferenz	Hardwareunterstütze Absicherung gegen lokale Angriffe	Nur mit geeigneter Hardware möglich; Angreifbarkeit über Seitenkanäle

den haben. Auf diese Weise können ML-Algorithmen unter anderem resistent gegen Privatsphäreangriffe gemacht werden, die wie Membership Inference auf einzelne Individuen in Trainingsdatensätzen abzielen.

DP kann dabei an verschiedenen Punkten der Datenverarbeitung ansetzen: *Lokale DP* wird auf den Eingabe-Datensatz angewendet, während *globale DP* die Ausgabe eines Algorithmus verändert und *algorithmische DP* in die Zwischenergebnisse von iterativen Algorithmen eingreift.

Ein Beispiel für algorithmische DP im ML-Kontext haben Abadi u. a. [1] vorgestellt. Ihr Ansatz fügt während des Trainings von neuronalen Netzen der Lernfunktion (hier: *stochastic gradient descent*) in jeder Iteration gezielt Rauschen hinzu, um DP zu erreichen [1].

Lokale DP kann insbesondere in verteilten ML-Szenarien wie *Federated Learning* genutzt werden. Dies ermöglicht es den teilnehmenden Parteien, ihre Daten lokal mit Privatsphäregarantien zu verändern. Anschließend können die so geschützten Daten im Trainingsprozess von Federated Learning (siehe folgender Abschnitt) verwendet werden [32].

DP bringt drei wichtige Eigenschaften mit sich: Einerseits ist DP *kompositionsfähig* (engl. *composable*), sodass die Komposition mehrerer DP-Mechanismen weiterhin *differentially private* ist. Andererseits ist DP robust gegenüber Hintergrundinformationen: Die Privatsphäregarantie von DP ist unabhängig vom möglichen Hintergrundwissen von Angreifenden, sodass ein Kombinieren der Algorithmenausgaben mit anderen Datenquellen keinen Erkenntnisgewinn aus Sicht der Angreifenden ermöglicht. Außerdem bietet DP Gruppenprivatsphäre (engl. *group privacy*), sodass korrelierte Eingaben (beispielsweise mehrere Datensätze zu einer Person) die Privatsphäregarantien nicht übermäßig abschwächen [1, 10].

Allerdings kann die Anwendung von DP in ML-Algorithmen dahingehend problematisch sein, dass die ohnehin schwer erklärbaren Prozesse des ML durch DP an zusätzlicher Komplexität gewinnen und für Menschen schwerer nachvollziehbar werden. Außerdem sind die Anwendungsmöglichkeiten mancher DP-Techniken auf bestimmte Datentypen beschränkt. Während die Reihenfolge tabellarischer Daten problemlos neu gemischt werden kann, ist dies für die Pixelreihenfolge auf Bildern nicht ohne Weiteres möglich. Obwohl bei Bilddaten geringfügiges Verrauschen, eine gängige Praxis zum Erreichen von DP, technisch ohne Weiteres möglich ist, sind die Auswirkungen von verrauschten Bilddaten für ML-Algorithmen mitunter unklar: Einerseits gibt es Angriffe gegen ML-Modelle mit sogenanntem *adversarial noise* (etwa: „bösartiges Rauschen“) in Trainingsdaten, wodurch Angreifende den Trainingsprozess bzw. das Modellverhalten zu ihren Gunsten manipulieren können [14]. Andererseits wurde gezeigt, dass das Hinzufügen von Rauschen in Trainingsdaten als Regularisierungsmethode eingesetzt werden kann, wodurch ein Modell weniger zur Überanpassung neigt, die ein häufiges Problem im Bereich des ML darstellt [39]. Vor einem großflächigen Einsatz von DP, etwa im Bereich der medizinischen Bildanalyse, sollte daher noch weitere Forschung betrieben werden [20].

Federated Learning

Sollen ML-Modelle mithilfe von Daten aus verschiedenen Quellen trainiert werden, so kann dies auf verschiedene Weisen erfolgen. Der klassische Ansatz besteht in der zentralen Zusammenführung aller verteilt vorliegenden Daten und dem anschließenden Trainieren auf dem so entstandenen Gesamtdatensatz. Auf diese Weise erhält die das Training durchführende Partei jedoch volle Kenntnis über die Daten aller Quellen. Handelt es sich um vertrauliche oder sensible Daten, so kann die Weitergabe an Dritte unerwünscht sein und im Zweifel dazu führen, dass die Daten nicht verwendet werden können. Durch *Federated Learning* (FL) können ML-Modelle mit Daten aus verschiedenen Quellen trainiert werden, ohne dass diese zuvor zusammengeführt werden müssen. Dabei findet das Training des Modells unter Nutzung lokaler Daten direkt auf den Geräten der Datenbesitzenden statt. Die Daten müssen somit weder mit anderen Teilnehmenden noch mit der zentralen, orchestrierenden Einheit geteilt werden. Es werden lediglich Mo-

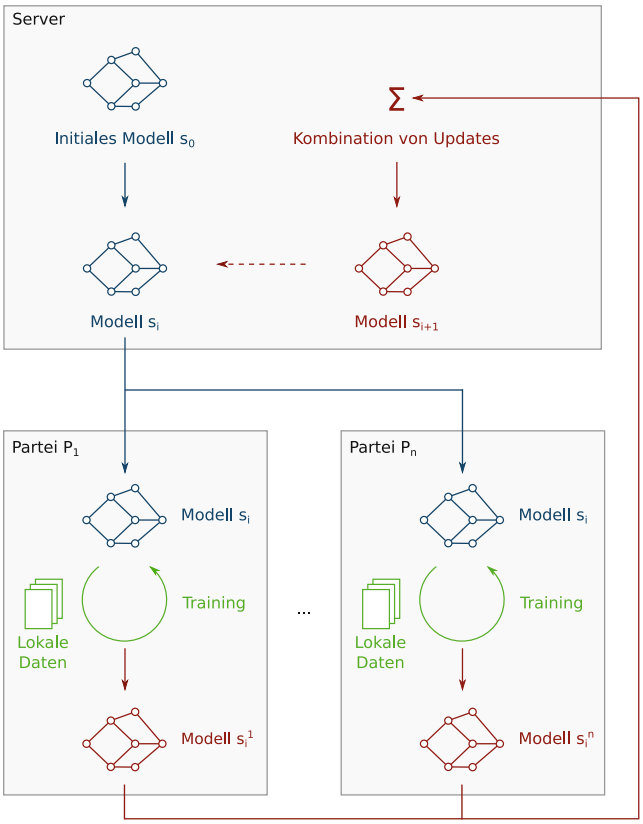

Abb. 5 Eine vereinfachte Darstellung des Federated-Learning-Prozesses für eine Runde i mit einem zentralen Server und n teilnehmenden Parteien

dellupdates zwischen den Teilnehmenden und einem zentralen Server ausgetauscht, der die Updates kombiniert und Runde für Runde das globale ML-Modell optimiert.

Die Trainingsphase für FL funktioniert dabei wie folgt: Der Server initialisiert zunächst ein ML-Modell s_0. Der Server wählt nun für jede Trainingsrunde i eine Menge von n Parteien aus, die in dieser Runde am Training des Modells beteiligt sind. Diese Parteien laden das aktuelle Modell s_i vom Server und trainieren s_i mit lokalen Daten weiter, wodurch das aktualisierte Modell s_i' entsteht. Das Modell s_i' (in der Praxis meist lediglich die Differenz zu Modell s_i) wird zurück an den Server geschickt, auf dem das Update s_i' mit den Updates aller anderen Parteien der aktuellen Trainingsrunde zu s_{i+1} kombiniert wird. Dies kann etwa durch die Berechnung des Durchschnitts aller Updates geschehen. In der nächsten Runde wird nun das weiterentwickelte Modell s_{i+1} trainiert. Die Anzahl der Trainingsrunden kann variieren und wird vom zentralen Server festgelegt. Das Training kann (je nach Einsatzgebiet) effizienter sein, wenn die am Training beteiligten Parteien von Runde zu Runde variieren [23, 25]. Der Ablauf einer Runde in der Trainingsphase wird in Abb. 5 dargestellt.

Ein bekanntes Anwendungsbeispiel für FL ist *Gboard*, die Touch-Tastatur von Google [16]. Diese bietet unter anderem die Vorhersage der nächsten Wörter, basierend auf dem bereits eingegebenen Text. An der Modellentwicklung sind hunderte Millionen Android-Geräte beteiligt, die die Modellupdates jeweils nachts während ihres Akkuladevorgangs berechnen.

Auch wenn die Trainingsdaten der Teilnehmenden während des FL-Trainings nicht aggregiert oder weitergegeben werden, können unter Umständen sensible Informationen aus individuellen Modellupdates extrahiert werden. Das Vorgehen ähnelt hier den eingangs beschriebenen Angriffen [18]. Abhilfe kann beispielsweise gezieltes Verrauschen der Modellupdates mithilfe von *Differential Privacy* oder der Einsatz von *Secure Multiparty Computation* schaffen [4, 23].

Secure Multiparty Computation

Besonders in verteilten Szenarien kann *Secure Multiparty Computation* (SMPC) eine der Grundlagen für privatsphärefreundliches ML bilden [8]. SMPC erlaubt es mehreren Parteien $P_1, ..., P_n$, eine Funktion f auf ihren jeweiligen persönlichen Eingaben $x_1, ..., x_n$ gemeinsam zu berechnen. Im Zuge der Berechnung erlangt jedoch keine Partei P_i Zugriff auf Informationen, die über ihre eigene Eingabe x_i und das Ergebnis der Berechnung $y = f(x_1, ..., x_n)$ hinausgehen. Dieses Prinzip wird in Abb. 6 dargestellt.

Um dieses Ziel zu realisieren, kommen in SMPC-Protokollen üblicherweise verschiedene Techniken zum Einsatz. Ein Beispiel für *Secure Two-Party Computation* (2PC) sind sogenannte *Garbled Circuits* [38], die insbesondere in Szenarien eingesetzt werden können, in denen die Trainingsdaten auf zwei nicht kollaborierende Parteien verteilt sind [28]. So könnten etwa zwei Krankenhäuser ein Modell mit Patientendaten trainieren, ohne dabei die Daten selbst an Dritte weiterzugeben. Im Kontext von Federated Learning kann ein aus dem SMPC-Bereich stammendes *Secure-Aggregation*-Protokoll dabei helfen, die Modellupdates einzelner Geräte zu kombinieren, ohne dabei Einsicht in die individuellen Beiträge zu gewähren (vgl. *lokale Privatsphäre* im Abschnitt „Differential Privacy“) [4]. Dies kann die privatsphärefreundliche Nutzung von anonymisierten Daten, die z. B. bei der Nutzung von Smartphones entstehen, für das Training von ML-Modellen erleichtern.

Neben der Trainingsphase kann auch die Inferenzphase in verteilten Szenarien durch SMPC geschützt werden. So erlaubt etwa das Framework *MiniONN* [24] das Überführen beliebiger neuronaler Netze in SMPC-geschützte Netze für eine verteilte Inferenzphase.

Ein wichtiges Auswahlkriterium für ein geeignetes SMPC-Framework ist das Angreifermodell, vor dem geschützt werden soll. Grundsätzlich wird zwischen *honest-but-curious* (auch: *semi-honest* oder passiv) und *malicious* (aktiv) unterschieden: Während sich im *honest-but-curious*-

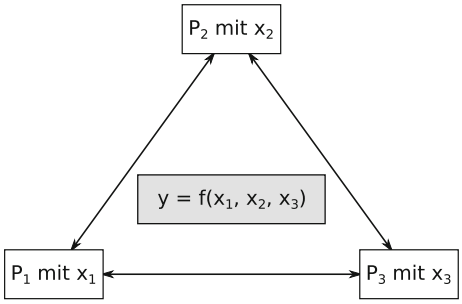

Abb. 6 Secure Multiparty Computation, hier dargestellt mit drei Parteien, erlaubt die Berechnung einer Funktion f auf den geheimen Eingaben x_i, ohne dass die Parteien die Eingaben anderer Parteien erfahren

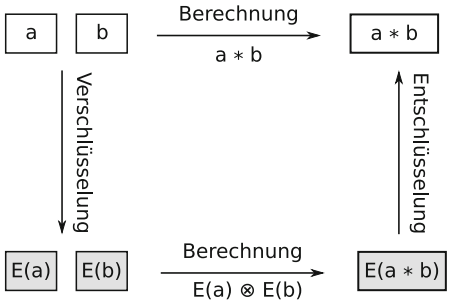

Abb. 7 Das Prinzip homomorpher Verschlüsselung, dargestellt anhand der Multiplikation zweier Zahlen a und b. $E(x)$ beschreibt den Schlüsseltext, der entsteht, wenn die Eingabe x verschlüsselt wird

Modell alle Teilnehmenden so verhalten, wie es das Protokoll vorschreibt und lediglich die Informationen nutzen, die ihnen preisgegeben werden, können *malicious* Angreifende auch vom Protokoll abweichen und so den Ablauf manipulieren. Die meisten implementierten Frameworks (z. B. *MOTION* [5]) schützen gegen das etwas schwächere *honest-but-curious*-Angreifermodell. Ein Schutz gegen aktive Angreifende ist in der Regel aufwendiger, wird aber beispielsweise im *MP-SPDZ*-Framework ermöglicht [21].

Beim Einsatz von SMPC muss ein besonderes Augenmerk auf die Recheneffizienz der eingesetzten Algorithmen gelegt werden. Eine achtlose Kombination von ML und SMPC kann ansonsten unverhältnismäßig hohe Rechenaufwände nach sich ziehen. Hierbei spielen insbesondere die Anzahl an beteiligten Parteien und die Komplexität der benötigten Berechnungen, etwa beim Training eines neuronalen Netzes, eine Rolle [6]. Einem produktiven Einsatz von SMPC steht dieser Aspekt derzeit häufig noch im Weg.

Homomorphe Verschlüsselung

In der Mathematik wird ein *Homomorphismus* als eine Abbildung verstanden, die die Elemente einer Menge auf eine andere Menge abbildet und dabei deren mathematische Struktur erhält. Daten, die per homomorpher Verschlüsselung (engl. *homomorphic encryption*, HE) verschlüsselt wurden, behalten ihre Struktur insofern, als dass auf ihrer verschlüsselten Form Berechnungen durchgeführt werden können, die Operationen auf den unverschlüsselten Daten entsprechen. Beispielsweise lässt sich so das verschlüsselte Produkt $E(a * b)$ zweier verschlüsselter Zahlen $E(a)$ und $E(b)$ als $E(a) \otimes E(b)$ berechnen, ohne die Daten entschlüsseln zu müssen. Die Verwendung unterschiedlicher Operationen $*$ und $\otimes$ für die Multiplikation zweier Zahlen in unverschlüsseltem bzw. verschlüsselten Zustand liegt darin begründet, dass es sich dabei – abhängig vom verwendeten HE-Schema – auch um unterschiedliche Operationen handeln kann. Dieser Zusammenhang wird in Abb. 7 dargestellt. HE wurde erstmals 1978 von Rivest u. a. [34] im Zusammenhang mit privatsphärefreundlichen Datenanalysen vorgeschlagen [34].

Im Kontext von ML kann HE für verschiedene Verfahren genutzt werden. Dabei können die jeweiligen Berechnungsschritte während der Trainings- und/oder Inferenzphase durch entsprechende HE-Operationen auf verschlüsselten Daten ersetzt werden. So haben Graepel u. a. [15] mit *ML Confidential* für verschiedene Klassifizierungsverfahren gezeigt, wie sowohl während der Trainingsphase mit homomorph verschlüsselten Daten ein Modell trainiert werden kann als auch in der Inferenzphase verschlüsselte Daten klassifiziert werden können [15].

Je nach verwendeter HE-Ausprägung sind die möglichen Rechenoperationen eingeschränkt: *Partially HE* erlaubt nur eine einzige Operation auf verschlüsselten Daten, beispielsweise die Multiplikation. Anwendungen mit *Somewhat HE* und *Leveled HE* ermöglichen verschiedene Rechenoperationen, beispielsweise Multiplikation und Addition, schränken jedoch die möglichen Gesamtberechnungen etwa in Form der maximalen Tiefe der sich ergebenden Schaltkreise ein. Die 2009 entwickelte *Fully HE* (FHE) [13] schränkt die Rechenoperationen grundsätzlich nicht ein. Bei jeder Operation werden die verwendeten verschlüsselten Daten allerdings etwas verrauscht, weshalb nach einigen Operationen sog. *Bootstrapping* (Neuberechnung und Wiederverschlüsselung) ausgeführt werden muss, um das Rauschen zurückzusetzen. Die Anwendungsmöglichkeiten von HE sind daher durch FHE zwar theoretisch gewachsen, aber das Bootstrapping bringt einen Rechenaufwand mit sich, der in vielen Fällen nicht praktikabel ist.

Werden andere HE-Schemata als FHE verwendet, müssen in bestimmten Fällen, etwa wenn neuronale Netze in Verbindung mit Aktivierungsfunktionenen wie ReLU und Sigmoid genutzt werden sollen, nicht-lineare Funktionen durch Polynome approximiert werden oder durch MPC evaluiert werden [3]. Beispiele hierfür zeigen Hesamifard u. a. [17], die ein neuronales Netz mit entsprechenden Ersatz-Aktivierungsfunktionen trainieren und mit *CryptoDL* ein Framework für HE-gestützte Inferenz entwickelt haben.

Eine Evaluation des Ansatzes zeigt lediglich minimale Performance-Einbußen durch das Ersetzen der Aktivierungsfunktionen und eine für viele Anwendungsfälle ausreichend schnelle HE-Inferenz [17].

Trusted Execution Environments

Die in den vorhergehenden Abschnitten diskutierten Angriffs- und Verteidigungsmöglichkeiten setzen alle beim Datenaustausch oder der Interaktion mit ML-Modellen an. Ein weiterer Angriffsvektor kann die Betriebssystemebene oder die Hardware betreffen, beispielsweise wenn im Rahmen von MLaaS private Daten auf virtuellen Maschinen in einer Cloud verarbeitet werden. Bösartige Cloud-Provider könnten den ausgeführten Code manipulieren oder sich mutwillig Zugriff auf die verarbeiteten Daten verschaffen.

Trusted Execution Environments (TEE, etwa „vertrauenswürdige Ausführungsumgebungen“, auch: *sichere Enklaven*) können in einem solchen Szenario zur Erreichung verschiedener Schutzziele beitragen. Einerseits können sie die Authentizität von ausgeführtem Code und die Integrität einer Laufzeitumgebung (inkl. des Arbeitsspeichers und CPU-Registern) garantieren, andererseits können sie die Vertraulichkeit des ausgeführten Codes und der verwendeten Daten gewährleisten [35]. Dafür sind spezielle Hardwarekomponenten nötig, die beispielsweise die Verschlüsselung von Bereichen im Prozessorspeicher erlauben. Die Sicherheitsgarantien von TEEs basieren auf dieser spezieller Hardware. Es gibt allerdings auch Angriffe, die ebenjene Hardware kompromittieren [9, 27].

Insbesondere in verteilten Szenarien können TEEs mit anderen PPML-Methoden kombiniert werden, um Vertrauen zu schaffen. So stellen Ohrimenko u. a. in [30] einen Ansatz vor, in dem Teilnehmende gemeinsam auf einem Server mit einer TEE in Kombination mit SMPC ein ML-Modell trainieren können. Der auszuführende Code kann dabei gegenseitig überprüft werden und in geschützte Bereiche des Server-Arbeitsspeichers übertragen werden. Auch die Trainingsdaten können verschlüsselt in die sichere Enklave übertragen werden, wo im Anschluss an das Training das Modell in verschlüsselter Form vorliegt und heruntergeladen werden kann.

Insgesamt bieten TEEs also schwierig zu kompromittierende Umgebungen für Code und Daten, die allein oder in Kombination mit anderen PPML-Verfahren für das Training von ML-Modellen oder auch in der Inferenzphase eingesetzt werden können.

Fazit

Durch die zunehmende Anzahl an Anwendungsmöglichkeiten von ML-Verfahren ist zu erwarten, dass der Einsatz von ML-Verfahren auch weiterhin an Bedeutung gewinnen wird. Die Gefahren für Privatsphäre und Sicherheit, die insbesondere bei der Verwendung sensibler Trainingsdaten entstehen können, müssen dabei stets beachtet werden. In dieser Artikelserie wurden im ersten Teil die oft abstrakten Begriffe rund um ML konkretisiert und gängige Verfahren zu dessen Nutzung erklärt. Im zweiten Teil wurden mögliche Privatsphäreangriffe im Kontext von ML aufgezeigt und Techniken für datenschutzfreundliches ML vorgestellt. Durch den geschickten Einsatz dieser Techniken können ML-unterstützte Fortschritte in sensiblen Bereichen wie der medizinischen Forschung erzielt werden, während das Risiko einer Gefährdung personenbezogener Daten reduziert werden kann. Da dem Einsatz vieler PPML-Techniken in der Praxis jedoch noch Hindernisse entgegenstehen, bleibt abzuwarten, wie sich dieses Feld in Wissenschaft und Praxis in den nächsten Jahren entwickeln wird.

Förderung Diese Publikation entstand im Rahmen des vom Innovationsfonds des Gemeinsamen Bundesausschusses geförderten, fakultätsübergreifenden Konsortialprojektes RABATT (01VSF18035). Ein Autor wird aus Mitteln des Bundesministerium für Wirtschaft und Klimaschutz im Rahmen des ZIM-Projektes PANDA (ZF4498402DH9) gefördert.

Funding Open Access funding enabled and organized by Projekt DEAL.

Literatur

1. Abadi M, Chu A, Goodfellow I, McMahan HB, Mironov I, Talwar K, Zhang L (2016) Deep learning with differential privacy. In: Proceedings of the ACM SIGSAC conference on computer and communications security
2. Ateniese G, Mancini LV, Spognardi A, Villani A, Vitali D, Felici G (2015) Hacking smart machines with smarter ones: How to extract meaningful data from machine learning classifiers. Int J Secur Networks 10(3):137–150
3. Boemer F, Cammarota R, Demmler D, Schneider T, Yalame H (2020) MP2ML: A mixed-protocol machine learning framework for private inference. In: Workshop on privacy-preserving machine learning in practice (PPMLP)

4. Bonawitz K, Ivanov V, Kreuter B, Marcedone A, McMahan HB, Patel S, Ramage D, Segal A, Seth K (2017) Practical secure aggregation for privacy-preserving machine learning. In: Proceedings of the ACM SIGSAC conference on computer and communications security
5. Braun L, Demmler D, Schneider T, Tkachenko O (2020) MOTION – a framework for mixed-protocol multi-party computation. IACR cryptology ePrint archive, report, Bd. 2020/1137
6. Cabrero-Holgueras J, Pastrana S (2021) SoK: Privacy-preserving computation techniques for deep learning. Proc Priv Enhanc Technol 2021(4):139–162
7. Choquette-Choo CA, Tramer F, Carlini N, Papernot N (2021) Label-only membership inference attacks. In: International conference on machine learning
8. Cramer R, Damgård IB et al (2015) Secure multiparty computation. Cambridge University Press, Cambridge
9. Du ZH, Ying Z, Ma Z, Mai Y, Wang P, Liu J, Fang J (2017) Secure encrypted virtualization is unsecure (arXiv 1712.05090)
10. Dwork C, McSherry F, Nissim K, Smith A (2006) Calibrating noise to sensitivity in private data analysis. In: Theory of cryptography conference
11. Fredrikson M, Jha S, Ristenpart T (2015) Model inversion attacks that exploit confidence information and basic countermeasures. In: Proceedings of the ACM SIGSAC conference on computer and communications security
12. Ganju K, Wang Q, Yang W, Gunter CA, Borisov N (2018) Property inference attacks on fully connected neural networks using permutation invariant representations. In: Proceedings of the ACM SIGSAC conference on computer and communications security
13. Gentry C (2009) Fully homomorphic encryption using ideal lattices. In: Proceedings of the forty-first annual ACM symposium on theory of computing
14. Goodfellow IJ, Shlens J, Szegedy C (2015) Explaining and harnessing adversarial examples (arXiv 1412.6572)
15. Graepel T, Lauter K, Naehrig M (2012) ML confidential: Machine learning on encrypted data. In: International conference on information security and cryptology
16. Hard A, Rao K, Mathews R, Ramaswamy S, Beaufays F, Augenstein S, Eichner H, Kiddon C, Ramage D (2019) Federated learning for mobile keyboard prediction (arXiv 1811.03604)
17. Hesamifard E, Takabi H, Ghasemi M (2016) CryptoDL: towards deep learning over encrypted data. In: Annual Computer Security Applications Conference (ACSAC)
18. Hitaj B, Ateniese G, Perez-Cruz F (2017) Deep models under the GAN: information leakage from collaborative deep learning. In: Proceedings of the ACM SIGSAC Conference on Computer and Communications Security
19. Hussain SS, Wang P (2022) PrivacyRaven – privacy testing for deep learning. https://github.com/trailofbits/PrivacyRaven. Zugegriffen: 27. Jan. 2022
20. Kaissis GA, Makowski MR, Rückert D, Braren RF (2020) Secure, privacy-preserving and federated machine learning in medical imaging. Nat Mach Intell 2(6):305–311
21. Keller M (2020) MP-SPDZ: A versatile framework for multi-party computation. In: Proceedings of the ACM SIGSAC Conference on Computer and Communications Security
22. Leino K, Fredrikson M (2020) Stolen memories: Leveraging model memorization for calibrated white-box membership inference. In: USENIX Security Symposium, S. 1605–1622
23. Li T, Sahu AK, Talwalkar A, Smith V (2020) Federated learning: Challenges, methods, and future directions. IEEE Signal Process Mag 37(3):50–60
24. Liu J, Juuti M, Lu Y, Asokan N (2017) Oblivious neural network predictions via minionn transformations. In: Proceedings of the ACM SIGSAC conference on computer and communications security
25. McMahan B, Moore E, Ramage D, Hampson S, y Arcas BA (2017) Communication-efficient learning of deep networks from decentralized data. In: Artificial intelligence and statistics
26. Melis L, Song C, De Cristofaro E, Shmatikov V (2019) Exploiting unintended feature leakage in collaborative learning. In: IEEE symposium on security and privacy (SP)
27. Moghimi A, Irazoqui G, Eisenbarth T (2017) Cachezoom: How sgx amplifies the power of cache attacks. In: International conference on cryptographic hardware and embedded systems
28. Mohassel P, Zhang Y (2017) SecureML: A system for scalable privacy-preserving machine learning. In: IEEE symposium on security and privacy (SP)
29. Murakonda SK, Shokri R (2020) ML privacy meter: Aiding regulatory compliance by quantifying the privacy risks of machine learning (arXiv 2007.09339)
30. Ohrimenko O, Schuster F, Fournet C, Mehta A, Nowozin S, Vaswani K, Costa M (2016) Oblivious multi-party machine learning on trusted processors. In: 25th USENIX Security Symposium
31. Papernot N, McDaniel P, Goodfellow I, Jha S, Celik ZB, Swami A (2017) Practical black-box attacks against machine learning. In: ACM Asia conference on computer and communications security
32. Pihur V, Korolova A, Liu F, Sankuratripati S, Yung M, Huang D, Zeng R (2018) Differentially-private "draw and discard" machine learning (arXiv 1807.04369)
33. Rigaki M, Garcia S (2020) A survey of privacy attacks in machine learning (arXiv 2007.07646)
34. Rivest RL, Adleman L, Dertouzos ML et al (1978) On data banks and privacy homomorphisms. Found Secur Comput 4(11):169–180
35. Sabt M, Achemlal M, Bouabdallah A (2015) Trusted execution environment: what it is, and what it is not. In: IEEE Trustcom/BigDataSE/ISPA
36. Shokri R, Stronati M, Song C, Shmatikov V (2017) Membership inference attacks against machine learning models. In: IEEE symposium on security and privacy (SP)
37. Wu X, Fredrikson M, Jha S, Naughton JF (2016) A methodology for formalizing model-inversion attacks. In: Computer security foundations symposium (CSF), S 355–370
38. Yao AC (1986) How to generate and exchange secrets. In: IEEE symposium on foundations of computer science
39. You Z, Ye J, Li K, Xu Z, Wang P (2019) Adversarial noise layer: Regularize neural network by adding noise. In: IEEE international conference on image processing (ICIP)
40. Zhang Y, Jia R, Pei H, Wang W, Li B, Song D (2020) The secret revealer: Generative model-inversion attacks against deep neural networks. In: IEEE/CVF conference on computer vision and pattern recognition