Ziele

Inhaltlicher Leitfaden für die gesamte Bachelorarbeit. Zu fully-, semi- und partial homomorphic encryption je eine umgesetzte Realisierung recherchieren und überlegen wie man sie klassifizieren könnte.

Für jede identifizierte und untersuchen, ob die Malleability des verwendeten Kryptosystems betrachtet wurde und wie damit umgegangen wurde. Bzw. ob integritätssichernde Maßnahmen ergriffen werden.

Fragen beim Lesen der Paper

1. Handelt es sich um ein fully-, semi-, oder partial homomorphic Kryptosystem?
2. Warum haben sich die Wissenschaftler für das jeweilige Kryptosystem entschieden?
3. Wurde malleability des Kryptosystems beachtet?
4. Wurden integritätssichernde Maßnahmen ergriffen? (vgl. „Integritätsprüfung homomorpher Operationen“)

Kurzdefinitionen

Malleability

Malleability beschreibt die Möglichkeit, dass ein Angreifer einen Chiffretext c von Klartext k geziehlt verformen kann um einen daraus abgeleiteten Chiffretext $f(c) = c'$ zu erzeugen welcher in einer ihm bekannten Beziehung f zu c steht. Existiert nun zwischen den Klartexten k und k' eine Beziehung die der Angreifer umkehren kann, kann er zu k' den ursprünglichen Klartext k bestimmten. [Sma03, p. 292]

Eigenschaften: Ein malleable Kryptosystem ist angreifbar mit chosen ciphertext Angriffen. (CCA2)

Kommentar: Das zwischen den Chiffretexten und den Klartexten eine ähnliche Beziehung steht die der Angreifer kennt ist eine Eigenschaft die genau Isomorphismen ermöglichen! Daher sind homomorphe Kryptosysteme per Design anfällig für malleability.

Homomorphic Encryption

Fully Homomorphic Encryption

Allows arbitrary computation on ciphertexts. (example: Gentry). It provides a ring structure (R,+,*) which allows to construct NAND gates. Since it is possible to create any boolean computation with NAND gates, a fully homomorphic cryptosystem allows to compute any arbitrary computation.

Semi-/Partial Homomorphic Enryption

Allows the homomorphic computation of some operations on ciphertexts (examples: ElGamal for multiplication, Pailler for addition). It can provide for example a Monoid (G,*,e) or an abelian group (G,+,e).[1]

[1] https://security.stackexchange.com/a/98190

Beispielkategorisierungen

Autocrypt

[TSCS13] Problemstellung: Server sind ständig durch Angriffe bedroht die bis hin zu ihrer kompletten Übernahme geraten können. Um Datendiebstahl und Vertraulichkeitsverletzungen vorzubeugen ist es ratsam nur mit verschlüsselten Datenbeständen zu arbeiten. Ein IT-System oder Programm so anzupassen, dass es auf verschlüsselten Daten korrekt arbeitet wollen die Wissenschaftler automatisieren indem sie die Arbeit der Programmtransformation mit einem Compiler abwickeln: Autocrypt.

Der Server läuft als virtuelle Maschine und Inhalte werden außerhalb der unvertrauten VM auf einem keyserver verschlüsselt. Autocrypt bestimmt automatisch benötigte Verschlüsselungsdatentypen für die Variablen und konvertiert zwischen diesen im Programmablauf her durch einfügen von hypercalls. Die Verschlüsselungsdatentypen werden gewählt nach der Verknüpfung die sie zu Verfügung stellen. Wenn also im Ursprungscode Additionen von zwei Variablen durchgeführt werden, dann werden zunächst Paillerverschlüsselungsdatentypen erzeugt. Wird das Ergebnis allerdings später multipliziert, dann muss der Datentyp konvertiert werden zu einem Elgamalverschlüsselungsdatentyp.

Bei der Entwicklung von Autocrypt sollen alle Rechenoperationen privacy-preserving sein. Als Transformationstool ist eine Integrität der Daten auf denen gerechnet wird daher nicht berücksichtigt worden.

Kategorisierungskriterien: Pailler wurde wegen seiner Homomorphie und Additionsverknüpfung verwendet (→ additiv-homomorph). Analoges Argument für Elgamal. Zwischen diesen beiden Verfahren wird hin und her konvertiert, da dies schneller ist als *ein* vollhomomorphes Verfahren (→ Klasse schneller Verfahren). Weiter ist Pailler flexibel einsetzbar für die Addition von Zahlen byteweiese oder bitweise. Letzteres ermöglicht die Konstrution eines homomorphen XOR Operators. (→ homomorph XOR)

Malleability: Die Autotoren haben als Zielsetzung die unerlaubte Kenntnisnahme von Daten auf dem Server zu unterbinden. Eine Überprüfung der Integrität von Rechenoperationen der von Autocrypt konvertierten Programmbestandteile ist daher kein Fokus der Arbeit [p. 4].

Machine Learning Classification over encrypted data

[BPTG15] Problemstellung: Es soll ein privacy-preserving Maschinenlernenverfahren erstellt werden, bei dem sowohl die zu klassifizierenden Daten als auch die Klassifiziererdesigns vertraulich bleiben. Es wird eine Bibliothek konstruiert, aus der Modular beliebige privacy-preserving Klassifizierer erstellt werden können.

In einem ersten Ansatz wurde überlegt privacy-preserving mit Secure Multiparty Computation umzusetzen, welches sich jedoch als zu langsam herrausgestellt hat. Aus dem gleichen Grund wird auch auf den Einsatz von vollhomomorpher Verschlüsselung verzichtet. Es ist schneller mit für Klassifizierungsverfahren spezialisierten Protokollen zu arbeiten.

Es wird wie in [TSCS13] XOR mit Pailler simuliert. Zusätzlich wird ein privates Skalarprodukt auf Basis von Pailler berechnet. Gegeben seien die Vektoren $x = (x_1, \ldots, x_n)$ und $y = (y_1, \ldots, y_n)$ wobei alle Einträge Klartexte sind. Das mit pub Paillierverschlüsselte Skalarprodukt ist dann:

$$Enc_{pub}(\langle x, y \rangle) = \prod_i Enc_{pub}(y_i)^{x_i} \bmod N^2 \tag{1}$$

Eine weitere Tatsache die im Paillerkryptosystem ausgenutzt wird ist der Klartextraum ungefähr 2^{1024} bit ist. Anstelle von lediglich Integern können Floatzahlen mit Pailler verschlüsselt werden, wenn man die IEEE 754 floating point Darstellung verwendet welche große Exponenten benötigt.

In dieser Arbeit wurde auch eine leveled vollhomomorphe Verschlüsselung (HELib) verwendet, jedoch der Umfang und die Gründe dafür bleiben ohne nähere Erläuterung [p. 4].

Kategorisierungskriterien: Es wurden die Kryptosysteme von Paillier und Goldwasser–Micali verwendet. Beide Aufgrund ihrer schnelleren Performance und der mathematischen Verknüpfung sie anbieten. (→ additiv-homomorph) (→ xor-homomorph). Analog zur unverschlüsselten Konstruktion von Gleitkommazahlen aus ganzen Zahlen kann mit Pailler ein Operator für die homomorphe Addition von Gleitkommazahlen konstruiert werden. (→ floatingpoint-additiv-homomorph)

Malleability: Es werden die homomorphen Kryptosysteme ledglich zum Rechnen im Chiffreraum verwendet. Ein Angriff der Malleability ausnutzt wird nicht betrachtet. Dies ist nachvollziehbar, da hier ein deterministischer Algorithmus abgearbeitet wird.

Privacy Preserving Matrix Factorization

[NIW+13] Bei der Generierung von userspezifischen Empfehlungen anhand vorheriger Wahlen eines Users ist Matrizenfaktorisierung ein weit verbreitetes Verfahren. Um dieses privacy-preserving zu machen soll ein System designt werden, welches Empfehlung geben kann ohne die Userbewertungen zu lernen.

Bei dem Design wird aus Performancegründen hash-ElGamal verwendet um verschlüsselte Wertungen zu maskieren für die Einheit, welche im Besitz des privaten Schlüssels ist. In dem Design bekommt das Recommendersystem (RecSys) vom User ein mit dem öffentlichen Schlüssel von Cryptoserviceprovider (CSP) verschlüsseltes Rating c. Damit der CSP dies Rating nicht aufdecken kann, addiert RecSys einen zufälligen Wert μ auf das Rating. CSP erhält $c' = c + \mu$.

Kategorisierungskriterien: Es wurde hash-ElGamal verwendet wegen seiner schnelleren Performance gegenüber Paillier und seiner Additivität (→ additiv-homomorph)

Malleability: Bei dem Design wird von einem honest-but-curious Angreifer ausgegangen. Also könnte RecSys aus Neugierde $\mu = 0$ addieren und so CSP ermöglichen alle Userratings zu lernen. Im HBC-Modell dürfen RecSys und CSP jedoch nicht vom Protokoll abweichen und könnten daher nicht kooperativ diese Information abschöpfen, denn CSP weiß nicht, dass RecSys eine Nulladdition durchführt welches die Maskierung aufhebt.

Kryptosystem	hom. Operator	sim. Operator	modi
Pailler	$+$	XOR, $\langle \cdot, \cdot \rangle$	bitwise, bytewise, float
ElGamal	$\cdot$	-	-
hash-ElGamal	XOR	-	-
Goldwasser-Micaeli	XOR		
BGV (HELib)	vollhom.	any	

Struktur der Bachelorarbeit

1. Einleitung (Was ist das Ziel? Wieso brauchen wir diesen Survey?)
2. Theoretische Grundlagen
2.1 Homomorphe Verschlüsselung
2.1.1 Semihomomorphe Verschlüsselung (verschiedene Vorstellen)
2.1.2 Fullyhomomorphe Verschlüsselung
2.2 Malleability
2.3 Privacy-Preserving
2.4 Ununterscheidbarkeit von Geheimtexten (Ciphertext Indistinguishability)
3. Klassifikation Homomorpher Verschlüsselung
3.1 Aufteilungen
3.2 Kategorisierung von Autocrypt...
4. Zusammenfassung

Literatur

[BPTG15] Raphael Bost, Raluca Ada Popa, Stephen Tu, and Shafi Goldwasser. Machine learning classification over encrypted data. In *NDSS*, 2015.

[NIW+13] Valeria Nikolaenko, Stratis Ioannidis, Udi Weinsberg, Marc Joye, Nina Taft, and Dan Boneh. Privacy-preserving matrix factorization. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 801–812. ACM, 2013.

[Sma03] Nigel Paul Smart. *Cryptography: An Introduction*, volume 5. McGraw-Hill New York, 2003.

[TSCS13] Shruti Tople, Shweta Shinde, Zhaofeng Chen, and Prateek Saxena. Autocrypt: enabling homomorphic computation on servers to protect sensitive web content. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 1297–1310. ACM, 2013.