Wissenschaftliche Ausarbeitung

PHOABE: Policy-Hidden Outsourced Attribute Based Encryption

Florian Hansen
Hochschule Flensburg

22. April 2020

Zusammenfassung

1 Einleitung

Diese wissenschaftliche Ausarbeitung wird im Rahmen der Hochschulveranstaltung *Hot-Topics in der IT-Security* angefertigt und handelt über den sicheren Datenaustausch zwischen Internet-of-Things-Geräten. Besonders wird hierbei auf die Attribute Based Encryption (ABE) eingegangen, welche ein relativ junges Forschungsgebiet der Kryptographie darstellt. Es existieren zwar einige ABE-Schemata, diese sind jedoch für leistungsschwache Geräte eher ungeeignet.

Diese Ausarbeitung beschäftigt sich hauptsächlich mit der Arbeit von [1], welche ein neues attribut-basiertes Verschlüsselungsschema vorstellt: *PHOABE*. Dieses Schema widmet sich bekannten Problemen der Attribut-basierten Verschlüsselung, wie zum Beispiel das Verstecken der Zugriffsregelungen (*Access-Policies*), die sensitive Daten enthalten können. Zusätzlich fordert das Schema eine Auslagerung des Entschlüsselungsvorganges, welches aufgrund von bilinearen Abbildungen relativ rechenintensiv ist. Hierbei sollen sogenannte *Semi-trusted Cloud Server* den aufwändigen Teil der Berechnung übernehmen, damit auch leistungsschwache Geräte solche attribut-basierte Verschlüsselungen durchführen können.

2 Problembeschreibung

Das *Internet der Dinge*, *Internet of Things* (IoT), hat die Aufgabe, verschiedenste Hardwarekomponenten miteinander zu verbinden und damit einen Datenaustausch zu ermöglichen. Ein gutes Beispiel hierfür sind Smartphones und Sensoren, die ständig miteinander kommunizieren müssen bzw. sollen. Grundsätzlich können IoT-Anwendungen aufgrund ihrer Kommunikationsfähigkeit in zwei Bereiche eingeteilt werden. In Einzelanwendungen benö-

tigen man lediglich eine Autorität während es auch Anwendungsfälle gibt, die über unterschiedliche Bereiche bzw. Domänen kommunizieren müssen. Diese besitzen damit mehrere Autoritäten.

Beide Formen von IoT haben eins gemeinsam. Sie müssen einen sicheren Austausch von Daten und die Sicherung der Privatsphäre gewährleisten. Hierfür werden Daten verschlüsselt und mithilfe von Zugriffskontrollmechanismen nur bestimmten Autoritäten zur Verfügung gestellt. Genau deshalb ist die attribut-basierte Verschlüsselung ein attraktiver Kandidat, da sie Zugriffskontrolle (*Access-Control*) und Verschlüsselung miteinander verknüpft. Ein Problem dabei exisitiert dennoch. Dieses Verfahren verwendet bilineare Abbildungen, was vor allem in der Entschlüsselung viel Rechenaufwand bedeutet. Für leistungsstarke Desktop-Rechner mag dies weniger eine Rolle spielen, ist jedoch bezogen auf IoT ein großes Problem, denn dort kommunizieren in der Regel Geräte mit sehr eingeschränkten Ressourcen miteinander. Zudem erhöht sich der Rechenaufwand proportional zur Anzahl der Attribute. Das Problem liegt also darin, Daten sicher und effizient zwischen ressourcenarmen Geräten auszutauschen, ohne die Privatsphäre der Benutzer zu verletzen.

3 Motivation

Die attribut-basierte Verschlüsselung ist, wie in der Problembeschreibung bereits erwähnt, ineffizienter, je mehr Attribute verwendet werden. Besonders die Entschlüsselung wird durch die bilinearen Operationen extrem rechenaufwändig, was zur Folge hat, dass dieses Schema auf ressourcenärmeren Geräten und damit auf IoT nur bedingt anwendbar ist. Eine wichtige Frage, die sich also stellt ist, wie man ABE für eben solche Anwendungsfälle anwendbar gestalten kann.

In weiterführenden Arbeiten, so auch der von [4], wird ein neues Konzept in Verbindung mit attribut-basierter Verschlüsselung vorgestellt, um diese effizienter zu gestalten. Für die kostspielige Entschlüsselung ist nun nicht mehr alleine der Benutzer zuständig, sondern eine weitere Instanz. Ein halb vertrauenswürdiger Cloud-Server (*semi-trusted cloud server*, STCS), welcher

den rechenaufwändigen Teil der Entschlüsselung übernehmen soll. Damit der Server keinerlei Informationen über den Entschlüsselten Ciphertext erhält, soll dieser diesen lediglich zum Teil entschlüsseln. Anschließend wird das Ergebnis vom Benutzer selbst entschlüsselt. So soll garantiert werden, dass die rechenaufwändige Arbeit vom STCS und nicht vom Benutzer übernommen wird und trotzdem keinerlei Informationen über die eigentliche Nachricht an dem Server preisgegeben wird.

Eine weitere Überlegung von [1] ist folgende. Was ist, wenn der STCS das teilweise entschlüsselte Chiffrat fälscht? Es muss also zusätzlich die Möglichkeit bestehen, dass das Ergebnis der Entschlüsselung des STCS' überprüft werden kann. Einige vergangene Arbeiten haben sich dem Problem gewidmet, jedoch keine effiziente Lösung geboten oder sind inpraktikabel für IoT, da sie sich auf einer einzigen Autorität stützen. Da IoT jedoch hauptsächlich über mehrere Domänen kommuniziert, ist dies keine praktikable Lösung. Die Motivation hinter [1] ist also, ein ausgelagertes, verifizierbares multi-authority ABE-Schema zu entwerfen.

4 Grundlagen

4.1 Zugriffsregeln

Grundsätzlich werden Regeln für den Datenzugriff durch zwei Formate repräsentiert. Zum Einen durch boolsche Funktionen und zum Anderen durch sogenannte Linear Secret Sharing Schemes (LSSS). In diesem Abschnitt sollen beide Repräsentationsmöglichkeiten eingeführt werden.

Definition 1 (Zugriffsstrukturen [5]). *Sei* $\{P_1, P_2, ..., P_n\}$ *die Menge der beteiligten Parteien. Eine Sammlung* $\mathbb{A} \subseteq 2^{\{P_1,P_2,...,P_n\}}$ *ist monoton, wenn* $\forall B, C. \quad B \in \mathbb{A} \quad \wedge \quad B \subseteq C \implies C \in \mathbb{A}$. *Eine Zugriffsstruktur ist damit eine Sammlung* $\mathbb{A}$ *von nicht-leeren Untermengen von dem Universum* $\{P_1, P_2, ...P_n\}$. *Alle Mengen* $A \in \mathbb{A}$ *werden als authorisierte Mengen während die nicht in* $\mathbb{A}$ *vertetenden Mengen als unauthorisierte bezeichnet werden.*

Definition 1 kann dabei so interpretiert werden, als dass alle Obermengen jedes Elements $B \in \mathbb{A}$ verteten sein müssen. Im Folgenden soll ein Beispiel dies verdeutlichen.

Beispiel. *Sei* $\mathbb{U} = \{1, 2, 3, 4\}$ *ein Universum und* $\mathbb{A} \subseteq 2^{\mathbb{U}}$ *eine Zugriffsstruktur.*

Die Zugriffsstruktur $\mathbb{A} = \{\{1, 2\}, \{3, 4\}\}$ *ist nicht monoton, da das Element* $\{1, 2, 3\}$ *nicht vorhanden ist.*

Die Zugriffsstruktur $\mathbb{A} = \{\{3, 4\}, \{1, 3, 4\}, \{1, 2, 3, 4\}\}$ *ist monoton, da alle Obermengen der Elemente von* $\mathbb{A}$ *vorhanden sind.*

Definition 2 (Linear Secret-Sharing Schemes [5])**.** *Ein Linear Secret-Sharing Scheme (LSSS) über eine Menge von Parteien P ist linear, wenn*

1. *Die Anteile (shares) für jede Partei einen Vektor* $\vec{v} \in \mathbb{Z}_p^{n+1}$ *formen.*
2. *Eine Matrix M existiert, die l Zeilen und* $n + 1$ *Spalten enthält. Jede i-te Zeile aus M mit* $i \in \{1, ..., l\}$ *ist dann mit einer Partei* $x_i \in P$ *beschriftet. Wenn wir den Spaltenvektor* $v = (s, r_1, r_2, ..., r_n)$ *betrachten, wobei* $s \in \mathbb{Z}_p$ *das zu teilende Geheimnis ist und* $r_1, ..., r_n \in \mathbb{Z}_p$ *zufällig gewählt werden, dann ist* Mv *ein Vektor bestehend aus l Anteilen des Geheimisses s.*

Beispiel. *Gegeben sei eine LSSS-Matrix* $M \in \mathbb{Z}_2^{l \times n}$ *und die Abbilding* $\rho : \mathbb{N}^+ \times \mathbb{Z}_p^{l \times n} \to \mathbb{Z}_p^n$*, welche die i-te Zeile der Matrix M zurückgibt. Zudem werden den Zeilen Parteien* P_i *zugewiesen, sodass* $\rho(i, M)$ *den Anteil der jeweiligen Partei liefert.*

$$M = \begin{array}{c} P_2 \\ P_2 \\ P_1 \\ P_3 \\ P_4 \end{array} \begin{pmatrix} 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix}$$

Betrachten wir die Kombination $\{P_2, P_4\}$. Mithilfe der Abbildung ρ erhalten wir die dementsprechenden Zeilen.

$$\begin{aligned}\rho(0, M) &= \vec{v_1}_{P_2} = (1, 1, 0, 1)\\ \rho(1, M) &= \vec{v_2}_{P_2} = (0, 1, 1, 0)\\ \rho(4, M) &= \vec{v}_{P_4} = (0, 0, 1, 1)\end{aligned}$$

Um die Authentizität zu prüfen, muss die Summe der Zeilenvektoren $e = (1, 0, 0, 0)$ ergeben. Wir berechnen nun also die Summe der Vektoren

$$\vec{v_1}_{P_2} + \vec{v_2}_{P_2} + \vec{v}_{P_4} = (1, 2, 2, 2) \equiv (1, 0, 0, 0) \mod 2$$

und sehen, dass die Kombination $\{P_2, P_4\}$ authorisiert ist.

4.2 Bilineare Abbildungen

Definition 3 (Bilineare Abbildungen [5])**.** *Sei $\mathbb{P}$ die Menge aller Primzahlen und $\mathbb{G}$, $\mathbb{G}_T$ zwei multiplikative zyklische Gruppen mit einer Ordnung $p \in \mathbb{P}$. Sei g ein Generator von $\mathbb{G}$ und $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ eine bilineare Abbildung mit folgenden Eigenschaften.*

1. *Bilinearität: $\forall u, v \in \mathbb{G},\ \ \forall a, b \in \mathbb{Z}_p.\ \ e(u^a, v^b) = e(u, v)^{ab}$*
2. *Nicht-Entartung: $e(g, g) \neq 1$*
3. *Effizient Berechenbar: Die Gruppenoperation von $\mathbb{G}$ und die bilineare Abbildung e sind effizient berechenbar.*

Beispiel. *Sei $(G, +)$ eine Gruppe und $x, y, z \in G$. Sei $(G_T, *)$ eine multiplikative Gruppe und $e : G \times G \to G_T$ eine bilineare Abbilding. Dann gilt*
$e(3x, y) = e(x + x + x, y) = e(x, y) * e(x, y) * e(x, y) = e(x, y)^3 = e(x, 3y)$.

5 Attribute Based Encryption (ABE)

Im Gegensatz zur klassischen Public-Key-Verschlüsselung zielt die attributbasierte Verschlüsselung (ABE) darauf ab, ein Chiffrat für mehrere anstatt für nur einen Benutzer zu erzeugen. Dafür werden die privaten Schlüssel und Chiffrate der Benuzer (Parteien) als Zugriffsregeln bzw. Attribute ausgelegt. Ferner ist ein Benutzer in der Lage das Chiffrat mithilfe von Attributen oder festgelegten Regeln zu entschlüsseln. Grundsätzlich werden ABE-Schemata in zwei unterschiedliche Kategorien eingeteilt [1].

5.1 Key-Policy ABE

KP-ABE-Schemata verwenden Zugriffsregeln als private Schlüssel und berechnen ein Chiffrat basierend auf ausgewählte Attribute. Als Beispiel soll die Zugriffsregel $A \wedge B$ als Schlüssel dienen. Zudem wird ein Chiffrat mit dem Attribut $\{A\}$ erzeugt. Versucht man nun das Chiffrat zu entschlüsseln, so schlägt dies fehl, da die Zugriffsregel nicht erfüllt wird. Werden hingegen die Attribute $\{A, B\}$ verwendet, wird die Regel eingehalten und das Chiffrat kann entschlüsselt werden. Hieraus wird ein mögliches Problem sichtbar, denn mit einem solchen Schema lässt sich nicht kontrollieren welche Benutzer Zugriff besitzen sollen. Stattdessen erhalten alle Benutzer Zugriff, denen eine entsprechende Zugriffsregel zugewiesen wurde [2].

Ein Key-Policy-ABE-Schema besteht aus insgesamt vier Algorithmen [3].

1. $Setup\,(n) \rightarrow (PK, SK_M)$: Dieser Algorithmus nimmt als Eingabe den Sicherheitsparameter n und liefert als Ergebnis eine Menge bestehend aus öffentlichen Parametern PK, sowie einen Master-Secret-Key SK_M.

2. $KeyGen\,(\mathbb{A}, SK_M) \rightarrow SK$: Als Eingabe nimmt dieser Algorithmus die Zugriffsstruktur $\mathbb{A}$ und den Master-Secret-Key SK_M. Es wird ein privater Schlüssel SK zurückgegeben.

3. $Enc\,(m, A, PK) \rightarrow c$: Der Verschlüsselungsalgorithmus nimmt als Eingabe die Nachricht m, eine nicht-leere Menge von Attributen A und die

öffentlichen Parameter PK. Als Ausgabe wird ein Chiffrat c erzeugt.

4. $Dec\,(c, SK) \rightarrow \{m, \perp\}$: Dieser Algorithmus nimmt als Eingabe ein Chiffrat c und den privaten Schlüssel SK. Als Ausgabe wird die Nachricht m oder ein Fehler (dargestellt als $\perp$) erzeugt.

5.2 Ciphertext-Policy ABE

CP-ABE verwendet im Gegensatz zu KP-ABE Attribute als private Schlüssel für Benutzer [2]. Zudem werden Chiffrate mithilfe von Zugriffsregeln erzeugt, also genau engegengetzt zu dem Vorgehen bei KP-ABE. Nehmen wir an, wir erzeugen einen Schlüssel für Benutzer 1 bestehend aus den Attributen $\{A, B\}$ und einen Schlüssel für Benutzer 2 bestehend aus den Attributen $\{B\}$. Bei einem Chiffrat, welches unter der Zugriffsregel $A \wedge B$ erstellt wurde, ist Benutzer 1 dazu in der Lage, dieses zu entschlüsseln. Benutzer 2 jedoch nicht. Bei diesem Verfahren ist man also in der Lage zu entscheiden, welche Daten für welchen Benutzer zur Verfügung stehen sollen bzw. welche Daten entschlüsselt werden können.

Ein Ciphertext-Policy ABE-Schema besteht aus insgesamt vier Algorithmen [6].

1. $Setup\,(n) \rightarrow (PK, SK_M)$: Dieser Algorithmus nimmt als Eingabe den Sicherheitsparameter n und liefert als Ergebnis eine Menge bestehend aus öffentlichen Parametern PK, sowie einen Master-Secret-Key SK_M.

2. $KeyGen\,(A, SK_M) \rightarrow SK$: Als Eingabe nimmt dieser Algorithmus eine Menge von Attributen A und den Master-Secret-Key SK_M. Es wird ein privater Schlüssel SK zurückgegeben.

3. $Enc\,(m, \mathbb{A}, PK) \rightarrow c$: Der Verschlüsselungsalgorithmus nimmt als Eingabe die Nachricht m, eine Zugriffsstruktur $\mathbb{A}$ und die öffentlichen Parameter PK. Als Ausgabe wird ein Chiffrat c erzeugt.

4. $Dec\,(c, SK) \rightarrow \{m, \perp\}$: Dieser Algorithmus nimmt als Eingabe ein

Chiffrat c und den privaten Schlüssel SK. Als Ausgabe wird die Nachricht m oder ein Fehler (dargestellt als $\perp$) erzeugt.

5.3 Single-Authority

5.4 Multi-Authority

6 Policy-Hidden Outsourced ABE

7 Fazit und Ausblick

Literatur

[1] S. Belguith, N. Kaaniche, M. Laurent, A. Jemai, and R. Attia. PHOABE: Securely outsourcing multi-authority attribute based encryption with policy hidden for cloud assisted IoT. *Computer Networks*, (133):141–156, 2018.

[2] J. Bethencourt, A. Sahai, and B. Waters. Ciphertext-policy attribute-based encryption, 2007.

[3] V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. Cryptology ePrint Archive, Report 2006/309, 2006. https://eprint.iacr.org/2006/309.

[4] M. Green, S. Hohenberger, and B. Waters. Outsourcing the Decryption of ABE Ciphertexts. *USENIX Security Symposium*, 2011.

[5] R. Ostrovsky, A. Sahai, and B. Waters. Attribute-based encryption with non-monotonic access structures. Cryptology ePrint Archive, Report 2007/323, 2007. https://eprint.iacr.org/2007/323.

[6] B. Waters. Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. Cryptology ePrint Archive, Report 2008/290, 2008. https://eprint.iacr.org/2008/290.