13. Juli 2017 Studie · Digitale Identität · Privatsphäre & Krypto · 20 Min. Lesezeit

Rich Ende-zu-Ende Verschlüsselung

Archiviert. Dieser Inhalt wird nicht mehr aktualisiert. Er bleibt zur Information verfügbar.

Quelle https://technology2.a-sit.at/veroeffentlichungen/rich-ende-zu-ende-verschluesselung/

Diese Projekt diskutiert die Anforderungen an eine Lösung, welche den Authentifizierungsschritt von der Anwendung trennt.

Aus dem Projekt „Skytust“ zur Verschiebung des Schlüsselmaterials von der Applikation zu einem Applikations-externen Punkt wurde das Schlüsselmaterial von der lokalen Umgebung in typisch besser schützbare zentrale Umgebungen verlagert. Damit sind aber Authentifizierungsinformationen, die zur Autorisierung der Schlüsselnutzung verwendet werden, umso exponierter. Diese Informationen müssen von einer Applikation gesammelt und entsprechend weitergeleitet werden – die Applikation hat damit Zugriff auf diese sensiblen Daten. Eine adäquatere Herangehensweise wäre, diese Daten außerhalb der Applikation zu sammeln. Dazu ist ein Service zwischen Applikation und Schlüsselservice notwendig, was wiederum nach einer Ende-zu-Ende-verschlüsselten Kommunikation verlangt. Die Herausforderung ist nun, die Authentifizierungsdaten so zu sammeln und weiterzuleiten, dass der sammelnde Dienst die verschlüsselte Kommunikation nicht aufbrechen muss, dennoch aber Daten zur Kommunikation hinzufügen kann.

Dieser Bericht diskutiert nun die Anforderungen an eine Lösung, bietet einen Technologieüberblick, und präsentiert schließlich ein allgemeines Konzept. Das allgemeine Konzept wird mit einem praktischen Anwendungsfall untermauert. Im Rahmen dieses Projekts bleiben allerdings wichtige Probleme auch ungelöst, die einen funktionierenden Demonstrator verhindern. Nichts desto trotz schafft es das Projekt, die Motivation und existierende Bausteine besser zu verstehen und schließlich eine mögliche Lösung aufzuzeigen.

1. Einleitung

Ende-zu-Ende Verschlüsselung als Konzept ist schon lange bekannt. In letzter Zeit erfreut sich diese Technologie allerdings steigendem öffentlichen Interesse. Wo früher nur vereinzelte Services (wie zum Beispiel das offene Extensible Messaging and Presence Protocol (XMPP) mit der Off-The-Record (OTR) Erweiterung) dieses Feature bereitgestellt haben, folgen heute Größen wie WhatsApp und Skype. Neben diesen Größen gibt es aber weitere Anwendungen, die auf Ende-zu-Ende Verschlüsselung setzen. In

dynamischen Netzwerksstrukturen (wie sie zum Beispiel in Wireless Sensor Networks zu finden sind) kommen oft multi-hop Protokolle zum Einsatz, die Datensätze zusammenführen und dann weitersenden, ohne dass die Verschlüsselung aufgebrochen werden muss. Zwei Endpunkte können so sicher und energieeffizient kommunizieren, dazwischenliegende Knoten können die Inhalte nicht einsehen. Es ist bislang aber nicht möglich, dass in einer Ende-zu-Ende verschlüsselten Kommunikation von bestimmten Knoten zwischen den Endpunkten bestimmte Inhalte gezielt gelesen, geändert, hinzugefügt oder gelöscht werden können, ohne dass der oder andere Knoten alle Teile der Kommunikation einsehen können. In diesem Projekt wird anhand eines konkreten Anwendungsfalls erörtert, wie eine solche Technologie die Datensicherheit von Benutzern und Benutzerinnen von kryptographischen Applikationen verändern kann. Dieses Projekt setzt die Erkenntnisse der Projekte rund um das Thema Skytrust voraus: Schlüsseldaten werden auf einem eigenen Webservice gehostet, die Kommunikation dorthin wird Ende-zu-Ende verschlüsselt. Realisiert ist dies zurzeit mit TLS, da nur zwei Punkte miteinander kommunizieren. Die durch das Projekt Skytrust erreichte Verschiebung von kryptographischem Schlüsseldaten weg von der Applikation hin zu einem applikations-externen Punkt ist es gelungen, die Schlüsseldaten selbst aus dem Visier vieler Angriffe zu nehmen. An die Stelle der Schlüssel selbst sind Authentifizierungsdaten getreten, mit denen eine Schlüsselnutzung autorisiert wird. Diese Daten werden aber von der Applikation gesammelt und weitergeleitet. Obwohl die Nutzung von Schlüsseln auf applikations-externen Services durch Policies eingeschränkt werden kann und damit die Missbrauchsmöglichkeiten begrenzt werden, ermöglichen Authentifizierungsdaten immer noch legitime kryptographische Operationen - gestohlene Daten können daher immer noch ausreichend Schaden anrichten. Damit ziehen Authentifizierungsdaten die Aufmerksamkeit von Angriffen auf sich. Ziel dieses Projekts ist es nun, den nächsten logischen Schritt zu versuchen, nämlich auch die benötigten und bisher von der Applikation gesammelten Authentifizierungsdaten auszulagern. Sobald die Applikation selbst die Authentifizierungsdaten nicht mehr kennt und in weiterer Folge diese Daten von einem anderen Gerät gesammelt werden können, ist es einem Angreifer oder einer Angreiferin, der/die die Applikation oder das Gerät bereits kontrolliert, nicht mehr möglich, direkt an die Authentifizierungsdaten zu kommen. Damit werden weitere Angriffsvektoren ausgeschlossen und der Aufwand für einen erfolgreichen Angriff steigt weiter. Ziel dieses Projekts ist es nun, die Authentifizierungsdaten auf einem dritten Gerät zu sammeln und in die Ende-zu-Ende verschlüsselte Kommunikation einzupflegen. Dazu ersetzen wir TLS mit einer einfachen Target Encryption basierend auf klassischer Public Key Cryptography mit ein paar Features von Attribute-based Encryption (ABE). Unsere Lösung erhebt keinen Anspruch auf Vollständigkeit und praxistauglicher Sicherheit – vielmehr soll ein neuer Weg ausprobiert werden, der zukünftig relevant werden könnte.

2. Rich End-to-End Verschlüsselung

In klassischen Netzwerktopologien gibt es meist nur Verbindungen zwischen Endpunkten. IPSEC oder TLS reichen damit völlig aus, um die Sicherheit der Daten Ende zu Ende zu gewährleisten. Für dieses Projekt nehmen wir allerdings eine komplexere Netzwerktopologie an, die sowohl über ein Sender/Empfänger-Paar verfügt, als auch über einen oder mehrere zwischenliegenden Knoten. Die Topologie ist in Abbildung 1 dargestellt. IPSEC und TLS stoßen hier an ihre Grenzen; es können Knoten A und B bzw. Knoten B und C sicher kommunizieren, nicht aber A und C. Sichere Kommunikation zwischen A und C ist nur dann möglich, wenn Knoten B in seiner Funktionalität soweit reduziert wird, dass dieser die Protokollpakete nur weiterleitet. Damit kann B die Daten aber nicht mehr verarbeiten. Dies ist gleichzeitig auch genau der Anwendungsfall der Ende-zu-Ende Verschlüsselung.

In modernen Anwendungen gibt es aber Fälle, wo es sehr wohl Sinn machen kann, Daten zwischen den Endpunktknoten der Kommunikation zu verarbeiten. Dabei wird natürlich genau festgelegt, welche Teile der Daten denn für welche Knoten les- oder sogar änderbar sind. Wir ziehen für dieses Projekt das Skytrust Framework heran. Das Skytrust Framework ermöglicht es, kryptographische Schlüsseldaten von einem Gerät (zum Beispiel ein Smart Phone) auf einen Webservice auszulagern. Das Webservice kann nun kryptographische Methoden, die die ausgelagerten Schüssel verwenden, nach erfolgreicher Authentifizierung der Benutzerin bzw. des Benutzers über das Internet bereitstellen. Am Gerät treten aber damit die Authentifizierungsdaten an die Stelle der Schlüsseldaten – wer die Authentifizierungsdaten kennt, kann die ausgelagerten Schlüssel verwenden. Ziel ist es also, auch das Sammeln der Authentifizierungsdaten auszulagern. Damit wäre am Gerät weder Schlüsseldaten noch Authentifizierungsdaten vorhanden und die Komplexität eines Angriffs wächst. Um die Erhebung der Authentifizierungsdaten auszulagern, ist ein weiteres Gerät in der Kommunikationslinie zwischen den Endpunkten notwendig. In Abbildung 1 stellt *Knoten A* das Gerät dar, das die kryptographische Operation anfordert, *Knoten B* das Gerät, das die Authentifizierungsdaten erhebt und *Knoten C* den kryptographischen Service. Nach wie vor soll *Knoten B* keine Daten mitlesen können. Wenn nun *A* einen Auftrag an *C* schickt und *C* eine Authentifizierungsaufforderung an *A* zurückschickt, könnte *B* dieser Aufforderung nachkommen und damit die Authentifizierungsdaten von *A* fernhalten. Dazu muss es *B* aber ermöglicht werden, den relevanten Teil (in diesem Fall die Authentifizierungsaufforderung) zu sehen, ohne dabei die übrige Kommunikation zwischen *A* und *C* zu kompromittieren.

Abbildung 1 Andere Anwendungsfälle sind ebenfalls denkbar. Ein weiterer Anwendungsfall aus dem Feld von Skytrust ist zum Beispiel, dass Authentifizierungsdaten immer von einem Proxy zwischen Client und Schlüsselserver angegeben werden, ohne dass eine Authentifizierungsaufforderung ausgegeben wird. Dieser Anwendungsfall ist in der Skytrust-enabled Org interessant. Die Skytrust-enabled Org stellt in einem Unternehmenskontext zentral einen Schlüsselservice bereit. Dort werden sämtliche Schlüssel aller MitarbeiterInnen verwaltet. Authentifiziert werden die MitarbeiterInnen aber zum Beispiel mit Microsofts Active Directory. Somit erspart sich die Mitarbeiterin oder der Mitarbeiter eine doppelte Authentifizierung und die Systemadministration kann über Active Directory auch Schlüsselberechtigungen steuern.

3. Technologieüberblick

Mit einem definierten Projektziel diskutieren wir nun existierende Ansätze und bewährte Lösungen und untersuchen ihre Tauglichkeit für die Rich-End-To-End Verschlüsselung. Die gewählten Technologien bieten einen Überblick über den aktuellen Stand der Technik und stehen exemplarisch für ähnliche Technologien. Wir erheben keinen Anspruch auf Vollständigkeit. Viele erfolgreiche Lösungen basieren auf Key Agreement. Key Agreement ermöglicht jedem Teilnehmer, denselben kryptographischen Schlüssel zu erzeugen, ohne dass der Schlüssel selbst übertragen werden muss. TLS und auch OTR [1] sind Beispiele dieser Lösungen. Diese Protokolle sind gemacht für Punkt zu Punkt Kommunikation. Zusätzliche Punkte zwischen den Endpunkten werden nicht unterstützt. Um Authentifizierungsfunktionalität zu bekommen muss für TLS mindestens ein Punkt ein öffentliches, im besten Fall verifiziertes und beglaubigtes Zertifikat besitzen. Für OTR wird auf die Lösung des Socialist Millionaire Problem zurückgegriffen. Das heißt, es wird ein shared secret verglichen und dessen Gleichheit bewiesen, ohne dass das secret übertragen werden muss. Beide Protokolle verschlüsseln die Nutzdaten mit symmetrischen Schlüsseln die bei TLS bei jeder Session und bei OTR bei jeder Nachricht geändert werden. TLS ist dadurch gut geeignet für Übertragung großer Datenmengen. OTR hingegen ist für hohe Sicherheit ausgelegt. Alles in allem sind weder TLS noch OTR

geeignet für unseren Anwendungsfall. Das Setup ist aufwändig, es sind keine intermediates möglich und die Authentifizierung ist bei TLS für viele kurzlebige Knoten ineffizient und bei OTR nur mit pre-shared secrets zu erreichen. Eine weitere bewährte Technologie ist onion-routing, wie es im TOR-Netzwerk [2] zum Einsatz kommt. Dabei werden die Nutzdaten so verschlüsselt, dass nur der Zielknoten die Daten lesen kann. Weiters werden einem Knoten nur die für den Knoten selbst relevanten Routing-Informationen zugänglich gemacht. Realisiert wird dies mit Public-Key Kryptographie. Bevor das Datenpaket gesendet wird, wird eine Route durch das Netzwerk festgelegt. Dabei werden die Nutzdaten/Routing-Information zwiebelschalenartig (engl.: onion) verschlüsselt, sodass ein Knoten im TOR-Netzwerk nur die für ihn bestimmten Daten lesen kann (target encryption). Damit kann onion-routing an Knoten, die zwischen Sender und Empfänger stehen, Daten auslesen und könnte prinzipiell auch neue Daten einfügen ohne die restliche Kommunikation lesen oder ändern zu können. TOR selbst ist aber zu komplex und kritisch um dort Änderungen vorzunehmen. Damit bleibt TOR erste Wahl für Anonymisierung, ist aber so nicht geeignet für unseren Anwendungsfall. Onion-routing bietet allerdings einige Ideen für unseren Anwendungsfall. Eine Technologie aus dem Bereich der Wireless Sensor Networks ist signature chaining [3]. Signature chaining ermöglicht es, Vertrauen von einem Netzwerkknoten über eine Zahl von dazwischenliegenden Knoten zu einem anderen Knoten zu übertragen, ohne dass ein dazwischenliegender Knoten seine Rolle in der Vertrauensübertragung leugnen kann. Dazu gibt es mittlerweile eine Vielzahl an Varianten [4] [5] [6] [7] [8], die oft stark auf wireless sensor networks abgestimmt sind. Dadurch sind die Lösungen sehr effizient und haben wenig overhead. Neben vielen interessanten Ideen und Ansätzen fokussiert das signature chaining aber darauf, dass Inhalte nicht geändert werden können. Deshalb ist diese Technologie nur teilweise für das aktuelle Projekt geeignet. Neben den genannten Lösungen gibt es noch weitere Ansätze wie homomophic encryption [9], proxy signatures [10], oder transitive signatures [11]. Diese werden hier aber nicht behandelt, da sie für das vorliegende Projekt sehr wenig Bedeutung haben.

4. Eine Lösung

In diesem Kapitel wir eine Lösung des Problems präsentiert. Die Lösung bedient sich der Public Key Kryptosysteme, verwendet target encryption (wie TOR) und bedient sich an Ideen aus dem signature chaining. Es gibt eine Setup-Phase und eine Betriebsphase. Damit ergibt sich eine recht einfache Lösung mit deutlichem Demonstrationscharakter, deren Ziel es lediglich ist, erste Schritte in diese neue Richtung zu machen. In diesem Kapitel wird nun die Architektur im Detail skizziert, der Setup-Prozess beschrieben und schließlich der Betrieb erläutert. Das Kapitel schließt mit einer Diskussion über Einschränkungen die sich durch schwerwiegende offen gebliebene Probleme ergeben.

4.1. Architektur

Wie bei jeder Kommunikation gibt es auch hier einen Sender- und einen Empfängerknoten. Der Sender erstellt Daten, der Empfänger verarbeitet Daten. Der Empfänger kann dem Sender mit Daten antworten, die der Empfänger konsumiert. Sender und Empfänger verwenden einen Ende-zu-Ende verschlüsselten Kommunikationskanal.

erstellt/konsumiert Daten CRUD Daten verarbeitet Daten *Abbildung 2* Damit ist es möglich, die Verbindung über dazwischenliegende Knoten zu leiten ohne Datensicherheit einzubüßen. Der Unterschied zu klassischen Kommunikationstechnologien ist nun, dass die dazwischenliegenden Knoten Daten einfügen, lesen, ändern oder entfernen (CRUD – create, read, update, delete) können, sofern sie von Sender oder Empfänger dazu berechtigt werden. Jeder Knoten besitzt daher eine statische Identität, die gezielte

Berechtigung (mittels target encryption) möglich macht. Realisiert werden diese Identitäten durch Schlüsselpaare von Public-Key Kryptosystemen. Die Architektur ist beispielhaft für vier teilnehmende Knoten in Abbildung 2 illustriert.

4.2. Setup

Bevor das System verwendet werden kann, müssen einige Dinge vorberechnet werden. Erstens ist es notwendig, dass jeder Knoten ein Schlüsselpaar *K* erstellt. Dabei wird ein passender Schlüsselgenerator *KeyGen* (bei uns RSA) mit einer Zufallszahl *random* und einer Länge *length* gefüttert um einen öffentlichen Schlüssel *Kpub* und einen dazu passenden privaten Schlüssel *Kpriv* zu erhalten (siehe Formel 1). Mit diesem Schlüsselpaar kann sich der Knoten zukünftig bekannt machen und verschlüsselte Daten empfangen. Weiters werden einem Knoten *N* Metadaten *meta* und Attribute *attrx* zugeordnet (siehe Formel 2). Metadaten dienen als Patzhalter für Daten, die hier unbedeutend sind, Attribute können zum Beispiel sein, ob ein Knoten Daten vom Benutzer abfragen kann um beispielsweise Authentifizierungsdaten zu erfahren.

$K = (Kpub, Kpriv) \leftarrow KeyGen(random, length)$ (1) $N \leftarrow (K, meta, attr1, attr2, ..., attrm)$ (2) Der Empfänger braucht eine Tabelle, in der Knoten und deren Attribute gelistet sind. Das ermöglicht es dem Empfänger, geeignete Knoten auszusuchen, zu denen er gezielt Daten schicken kann (siehe Formel 3).

$Nodes(attr) \leftarrow \{N \in alle\ Knoten \mid attr \in Attrs(N)\}$ (3) Beispielsweise kann der Empfängerknoten so eine Authentifizierungsaufforderung entlang des Pfades zurückschicken und die Aufforderung an den ersten Knoten des Pfades adressieren, der in der Lage ist, Authentifizierungsdaten vom Benutzer abzufragen.

4.3. Betrieb

Nach dem Setup ist das System bereit für den Betrieb. Dabei kommunizieren Sender- und Empfängerknoten über einen oder mehrere dazwischenliegende Knoten. Um Ende-zu-Ende-Verschlüsselung zu gewährleisten, verschlüsseln Sender und Empfänger die Nutzdaten für den jeweils anderen mithilfe derer öffentlicher Schlüssel. Immer wenn ein Knoten ein Datenpaket sendet, muss dieses mit dem Schlüssel des Knotens signiert werden. Daraus ergeben sich unter anderem zwei wichtige Eigenschaften. Erstens wird damit sichergestellt, dass unterwegs keine Daten unerlaubt geändert oder gelöscht wurden. Weiterhin ist es aber möglich, weitere Daten hinzuzufügen. Damit kann ein Knoten beispielsweise ungefragt Authentifizierungsdaten einfügen. Anwendungsfälle sind hier zum Beispiel Webapplikationen mit HTTP-Authentifizierung oder Authentifizierungsserver in Unternehmen. Zweitens lässt sich so später der Pfad durch das Knotennetzwerk rekonstruieren. Damit weiß der Empfänger, welche Knoten in der Kommunikation teilnehmen. Sollte der Empfänger weitere Daten benötigen, kann er nun mittels Attributtabelle gezielt bei ausgewählten Knoten danach fragen, ohne dass andere Knoten die Kommunikation lesen können. Mit source routing kann der Empfänger auch sicher sein, dass das Paket bei den vorgesehenen Knoten ankommen wird. Wenn ein adressierter Knoten nun das Paket entgegennimmt, kann er das Paket auspacken, lesen, beantwortet zurückschicken oder dem Pfad entlang weiterleiten. Schickt der Knoten das Paket beantwortet zurück werden nachfolgende Knoten nicht mehr bemüht, was somit Zeit und Aufwand spart. Leitet der Knoten das Paket unbeantwortet weiter (beispielsweise, weil der Benutzer diesen Knoten gerade nicht bedienen kann), bekommt der nächste Knoten das Paket. Somit wird es beispielsweise möglich, Authentifizierungsdaten entlang des Kommunikationspfads abzufragen und damit die Daten erst gar nicht am Sendergerät zu haben.

4.4. Offene Probleme

Wie bereits erwähnt hat dieses Konzept Demonstrationscharakter. Damit bleiben die üblichen (meist gelösten) Probleme sicherer Kommunikation unbeachtet. Da in diesem Konzept die einfachsten Methoden asymmetrischer Kryptographie zum Einsatz kommen, können Probleme wie Replay-Attacken, Man-in-the-Middle-Attacken, Effizienz und andere leicht durch passendere Kryptographie adressiert werden. Es bleiben aber auch Probleme, die die Informationssicherheit beeinflussen und nicht einfach zu beseitigen sind. Eines dieser Probleme ist die Identifizierung/Authentifizierung von Knoten. Der klassische Ansatz, um Vertrauen herzustellen ist entweder eine Zertifizierungsstelle, die die Identität eines Knotens bestätigt, oder eine auf PGP basierende Vertrauensbildung durch eine Community. Diese beiden Varianten funktionieren gut für die Endpunkte der Kommunikation, da es für jede Kommunikation nur zwei dieser Knoten gibt und diese schon vor der Kommunikation bekannt sind. Für Knoten, die erst zur Laufzeit in die Kommunikation eingebunden werden ist eine Zertifizierung möglicherweise nicht geeignet. Ein Beispiel sind Knoten, die dynamisch erstellt werden und nur kurz existieren. Solche Knoten leben zum Beispiel in einem IFrame im selben Browser-Fenster wie der Quellknoten, werden dynamisch als JavaScript-Knoten instanziiert und verschwinden mit dem Schließen des Browser-Fensters wieder. Einerseits ist es solchen Knoten kaum möglich, einen privaten Schlüssel sinnvoll zu schützen und anderseits ist es kaum möglich einen Schlüssel - in der kurzen Lebenszeit des Knotens - mit Hilfe einer Zertifizierungsstelle zu bestätigen. Die Identifizierung bzw. Authentifizierung eines Knotens bleibt somit ein offenes Problem. Ein weiteres offenbleibendes Problem ist, eine kryptografische Bindung zwischen zwei Protokollpaketen zu erreichen (z B. einem Request und einer nachfolgenden angeforderten Authentifizierung). Fehlt eine solche Bindung, ist es weder dem Nutzer noch dem Service möglich zuzuordnen, welche Authentifizierung zu welchem Request gehört. Einfache Zeitstempel oder Identifikatoren halten Angriffen nicht stand. So könnte ein Angreifer einem Nutzer eine Authentifizierungsaufforderung senden, die der Nutzer nach einem Request möglicherweise erwartet, noch bevor der Service dies tun kann. Der Angreifer könnte so unauffällig und effektiv die Authentifizierungsdaten des Nutzers abgreifen und nachfolgend missbrauchen. Zuletzt sei noch erwähnt, dass das Zusammenstellen der Attributstabelle nicht trivial ist. Dazu muss der Service alle potentiellen Knoten kennen. Dies ist problematisch, da dynamisch erstellte, kurzlebige Knoten existieren können, oder ein Knoten aufgrund von Umwelteinflüssen gerade nicht erreichbar ist (ein Knoten auf einem mobilen Gerät). Ein zukünftiges Ziel ist es, so eine Attributstabelle gar nicht mehr zu benötigen. Mit dem hier vorgestellten Konzept ist das aber nicht möglich, da für die Target Encryption vorher bekannt sein muss, welcher Knoten ein Paket lesen können muss. Generell bleibt das vertrauenswürdige Attestieren von Attributen ein Problem, das weitere Aufmerksamkeit benötigt.

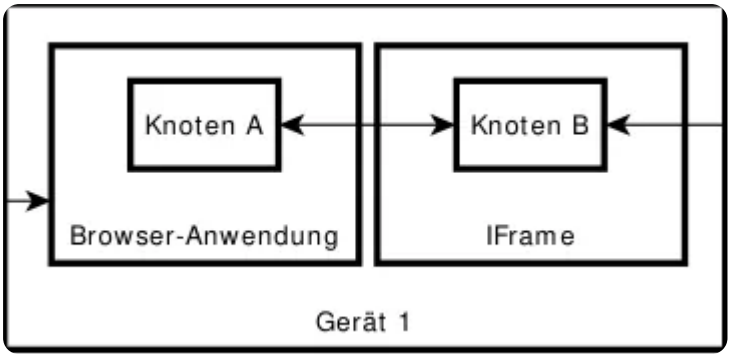
5. Praktischer Anwendungsfall

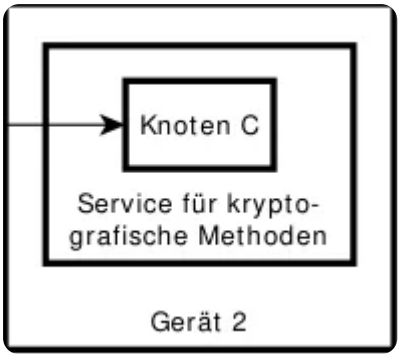
Der hier beschriebene praktische Anwendungsfall demonstriert, dass es mit dem vorhin vorgestellten Konzept gelingt, neben kryptografischen Schlüsseln auch die zugehörigen Authentifizierungsdaten aus einer Anwendung zu entfernen, damit die Angriffsfläche zu reduzieren und gleichzeitig aber eine nahezu unveränderte Benutzbarkeit zu erreichen. Um kryptografische Schlüssel vom Gerät auszulagern, wird CrySIL (ehem. Skytrust) verwendet. CrySIL verfügt bereits über ausreichende Funktionalität, um diesen Teilbereich erfolgreich zu realisieren. Wir erweitern CrySIL so, dass es auch möglich wird, die Authentifizierungsdaten abseits der Applikation zu sammeln. Wir demonstrieren diese Erweiterung anhand

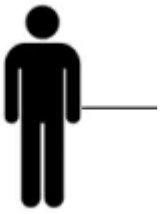
einer Browser-Applikation, die für ihre kryptografischen Bedürfnisse auf CrySIL zurückgreift. Der hier vorgestellte Anwendungsfall sammelt Authentifizierungsdaten auf demselben Gerät auf dem auch die Applikation läuft.

5.1. Architektur

Nachfolgend ist die Architektur des Demonstrators beschrieben. Eine Illustration dazu findet sich in Abbildung 3. Das Gerät des Benutzers/der Benutzerin (*Gerät 1* in Abbildung 3) betreibt zwei Knoten des CrySIL Frameworks. Die Browser- *Anwendung* verwendet W3C's Web Cryptography API, die direkt einen lokalen, dynamischen Knoten des CrySIL nutzt (*Knoten A* in







Col1	Knoten A Knoten B Browser-Anwendung IFrame Gerät 1	Col3	Knoten C Service für krypto- grafische Methoden Gerät 2
	Gerät 1 Browser-Anwendung IFrame Knoten A Knoten B		
r/	r/	r/	r/

Benutzerin *Abbildung 3* Abbildung 3), um kryptografische Methoden anzubieten. Aufgrund der bisher offen gebliebenen Probleme (Kapitel 4.4 erklärt die Hintergründe), verfügt der dynamische *Knoten A* über eine über die Instanzen konstantes Schlüsselpaar. Obwohl dies die Anzahl der Client-Instanzen der *Browser-Anwendung* auf eine einzige Instanz beschränkt, ermöglicht es dieses konstante Schlüsselpaar, die Identität des Client-Knotens zu kennen und damit den Demonstrator zu betreiben. *Knoten B* (in Abbildung 3) übernimmt als Teil von CrySIL die Aufgabe, Authentifizierungsdaten von der Benutzerin bzw. vom Benutzer abzufragen. Dieser Knoten wird mittels *IFrame* visuell im Browser-Fenster der Anwendung dargestellt, bleibt aber eine eigenständige Anwendung einer von der *Browser-Anwendung* unterschiedlichen Domain und wird damit vom Browser selbst nach Möglichkeit von der *Browser-Anwendung* isoliert. Nur gezielt verfügbar gemachte Kommunikation ist zwischen den *Knoten A* und *B* möglich. In unserem Demonstrator wird die Kommunikation mit Hilfe des HTML5-Messaging-Standards implementiert. Dieser Knoten kann also durch die in die *Browser-Anwendung* eingebettete Darstellung mit dem Benutzer oder der Benutzerin interagieren und dadurch sensitive Authentifizierungsdaten von der *Browser-Anwendung* fernhalten. Der *Service Provider*, der die Ausführung von kryptografischen Methoden übernimmt, lebt vollständig auf einem dem Gerät der Benutzerin bzw. des Benutzers verschiedenen *Gerät 2* . *Knoten C* (in Abbildung 3) als Teil von CrySIL kommuniziert wieder über eine definierte Schnittstelle mit den *Knoten A* und *B* . *Knoten C* ist im Gegensatz zu *A* und *B* eine Server-Instanz, deren Schlüsselpaar problemlos mittels Zertifizierungsstelle bestätigt werden kann. Dieser Knoten ist auch dafür zuständig, bei Bedarf Authentifizierungsanforderungen auszusenden und das Netzwerk an Knoten zu kennen. Im Demonstrationsfall wurden zumindest die Attributstabelle mit Hilfe der konstanten Schlüssel manuell erstellt. Alles in allem gleicht die Architektur mit den eben beschriebenen Änderungen, Eigenschaften und Definitionen dem Konzept, wie es in Kapitel 4 beschrieben wurde. Leider ist aufgrund der offen gebliebene Probleme aber kein voll funktionstüchtiges Setup möglich.

5.2. Protokoll

In diesem Kapitel wird nun zuerst das Protokoll beschrieben, das CrySIL verwendet. Danach werden die Änderungen aufgezeigt, die das Protokoll mit dem oben vorgestellten Prinzip kompatibel macht.

Das Protokoll von CrySIL wurde mit dem Hintergedanken konzipiert, dass später einfach intelligenteres Routing eingebaut werden kann. Dabei wurde auch Wert darauf gelegt, das Protokoll möglichst technologie-unabhängig zu gestalten. So kommunizieren im CrySIL Demonstrator zum Beispiel Browser-Fenster über Bytearrays, für die Kommunikation mit einem Webservice wird eine JSON Struktur verwendet. Alle haben gemeinsam, dass die Header Payload *Abbildung 4* Payload vom Header getrennt ist (siehe Abbildung 4). Der Header enthält Routing-Informationen, die am Kommunikationsweg geändert werden darf. Der Payload-Bereich enthält Nutzdaten, die nicht geändert werden dürfen. Nutzdaten können ein Verschlüsselungsbefehl genauso sein, wie eine Authentifizierungsaufforderung. Bisher wurden die Eigenschaften des jeweiligen Blocks nur definiert und nicht erzwungen. Die klare Trennung zwischen Daten, die geändert werden dürfen und jenen, die nicht Encrypted Encrypted Header Signature Signature Encrypted Payload Payload *Abbildung 5* geändert werden dürfen, wird für dieses Projekt ausgenutzt. Der Header bleibt wie er ist. Nach wie vor finden sich dort Daten, die zum Routing des Datenpakets benötigt werden. Der Payload-Teil wird nun aber verschlüsselt, sodass die Daten nicht von jedem gelesen werden können. Damit können Datenpakete von dritten Knoten weitergeleitet werden, ohne dass deren Inhalt bekannt wird. Dazu kommt eine Signatur, die einen veränderten Payload-Block erkennen lässt (siehe Abbildung 5). Somit können Datenpakete weder unerlaubt gelesen oder verändert werden. Will ein Knoten einer Kommunikation zusätzliche Daten bereitstellen, kann er einen weiteren Payload-Block erstellen. Jeder

Knoten muss, sollte er das Paket weiterleiten, das gesamte Datenpaket signieren. So kann nun der Pfad des Pakets rückverfolgt werden und zugleich ist dokumentiert, wer welche Daten geändert oder hinzugefügt hat. Mit diesen Änderungen ist auch das Protokoll von CrySIL so modifiziert, dass das oben erwähnte Konzept umgesetzt werden kann. Es bleiben die offenen Probleme, die das Konzept selbst hat.

5.3. Setup und Betrieb

Aufgrund der offen gebliebenen Probleme des Konzepts wurde auf eine Implementierung verzichtet. Einen funktionierenden Prototyp/Demonstrator gibt es daher nicht.

6. Future Work

Die wohl wichtigste zukünftige Arbeit behandelt die in Kapitel 4.4 beschriebenen offen gebliebenen Probleme. Ohne praktikable Lösungen zu diesen Problemen ist das Konzept nicht sicher umsetzbar. Weitere zukünftige Arbeiten sind nach Lösung der Probleme ein Demonstrator, der im Rahmen dieses Projekts nicht erstellt werden konnte. Weiters kann versucht werden, die Payload so zu markieren, dass nur die markierten Bereiche verändert werden können. Und zu guter Letzt gibt es auch andere Anwendungsfälle, die von dem oben vorgestellten Konzept profitieren können.

7. Zusammenfassung

In diesem Projekt wurde das Konzept der Rich End-To-End Encryption behandelt, ein Konzept, das es ermöglicht, in eine Ende-zu-Ende verschlüsselten Kommunikation kontrolliert und sicher Daten einzufügen. Der Bericht diskutiert eingangs, warum so eine Funktionalität gerade in der nahen Zukunft wichtig ist, definiert den Terminus der Rich End-To-End Encryption, zeigt einige Bausteine auf, die für eine Lösung des Problems in Frage kommen und präsentiert schließlich einen Lösungsansatz. Für eine vollständige Lösung bleiben allerdings zu viele schwerwiegende Probleme offen die einen erfolgreichen Demonstrator verhindern. Obwohl dieses Projekt nicht alle Ziele erreichen konnte, legt es doch eine gute Basis für das Verständnis der Materie und bietet damit eine gute Grundlage für zukünftige Arbeiten.

8. Literaturverzeichnis

[1] N. B. u. I. G. u. E. Brewer, „Off-the-record communication, or, why not to use PGP.," in *Proceedings of the 14th ACM Workshop on Privacy in the Electronic Society*, New York, NY, USA, 2004.

[2] L. L. a. R. S. a. m. Pease, „TOR: The Second-Generation Onion Router," in *Proceedings of the 13th USENIX Security Symposium*, 2004.

[3] A. S. u. B. Soh, „One-Way Signature Chaining - a New Paradigm of Group Cryptosystems,"

International Journal of Information and Computer Security, pp. 268-296, October 2008.

[4] A. B. u. C. G. u. A. O. u. D. H. Yum, „Ordered Multisignatures and Identity-based Sequential Aggregate Signatures, with Applications to Secure Routing.," in *Proceedings of the 14th ACM Conference on Computer and Communications Security*, New York, NY, USA, 2007.

[5] J. K. L. u. J. B. u. J. Zhou, „Certicate-based sequential aggregate signature," in *Proceedings of the Second ACM Conference on Wireless Network Security*, New York, NY, USA, 2009.

[6] H. Z. J. M. Q. L. a. J. X. Ximeng Liu, „Efficient attribute-based sequential aggregate signature for wireless sensor networks,“ *International Journal of Sensor Networks,* Bd. 16, Nr. 3, pp. 172-184, Jänner 2014.

[7] M. N. a. G. Tsudik, „Authentication of Outsourced Databases Using Signature Aggregation and Chaining,“ in *International Conference on Database Systems for Advanced Applications*, 2006.

[8] A. S. a. B. Soh, „A scalable wireless routing protocol secure against route truncation attacks,“

in *Proceedings of the 11th IFIP TC 6/TC 11 International Conference on Communications and Multimedia Security*, Berlin, 2010.

[9] M. N. u. K. L. u. V. Vaikuntanathan, „Can homomorphic encryption be practical?,“ in *Proceedings of the 3rd ACM workshop on Cloud computing security workshop*, 2011.

[10] M. M. u. K. U. u. E. Okamoto, „Proxy signatures for delegating signing operation,“ in *Proceedings of the 3rd ACM conference on Computer and communications security*, New York, NY, USA, 1996.

[11] S. M. u. R. L. Rivest, „Transitive signature schemes,“ in *CT-RSA*, 2002.