Karlsruhe Institute of Technology

KASTEL

Titel

Bachelorarbeit

KIT – Karlsruher Institut für Technologie
ITI – Institut für Theoretische Informatik
Forschungsruppe Kryptographie und Sicherheit

Yuguan Zhao

07. Juli 2017

Verantwortlicher Betreuer:	Prof. Dr. rer. nat. Jörn Müller-Quade
Betreuender Mitarbeiter:	Jeremias Mechler, M.Sc

Erklärung der Selbstständigkeit

Hiermit versichere ich, dass ich die Arbeit selbständig verfasst habe und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe, die wörtlich oder inhaltlich übernommenen Stellen als solche kenntlich gemacht habe und die Satzung des Karlsruher Instituts für Technologie zur Sicherung guter wissenschaftlicher Praxis in der gültigen Fassung beachtet habe.

Karlsruhe, den 07. Juli 2017

(Yuguan Zhao)

Inhaltsverzeichnis

Einleitung **3**

0.1 Motivation und Zielsetzung . . . 3

0.2 Grundlagen . . . 3

0.2.1 Vorbereitung . . . 3

0.2.2 Aufbau einer Email . . . 3

0.2.3 Signatur-Schema . . . 4

0.2.4 Commitment-Schema . . . 4

1 Sanitizable Signatures **7**

2 Aufbau von Hash Tree **9**

2.1 Schema CommitVector . . . 9

2.2 Schema HashTree . . . 9

3 Realisierung in Java **11**

3.1 Email Parser in Java . . . 11

3.2 Sanitizable Signatures in Java . . . 11

3.3 Schema HashTree in Java . . . 11

4 Vergleich von Ergebnissen **13**

5 Zusammenfassung und Ausblick **15**

Notation

Spezielle Bezeichner

H	Kollisionsresistente Hashfunktion
CH	Chameleon Hashfunktion
C	Ausgabe von Chameleon Hashfunktion
$m_1, ..., m_t$	Zusammensetzung einer Nachricht m
ID_m	Identifikationsnummer einer Nachricht m
g	Erzeuger einer zyklischen Gruppe von Primzahlordnung
q	Primzahlordnung einer zyklischen Gruppe
e	String-Hashwert bestimmter Länge einer Nachricht
σ	Signatur
r	Zufallszahl
b	Ein Bit
pk	Öffentlicher Schlüssel
sk	Geheimer Schlüssel
vk	Verifikationsschlüssel
$sigsk$	Signaturschlüssel

Einleitung

0.1 Motivation und Zielsetzung

E-Mail-Signaturen umfassen komplette Nachrichten, also insbesondere Body und Anhänge. Löscht man einen Anhang, beispielsweise um Speicherplatz zu sparen, kann die Gültigkeit der Signatur nicht mehr verifiziert werden. Ziel meiner Arbeit ist es, passende S/MIME Signaturverfahren, Standards für die Verschlüsselung und das Signieren von MIME-gekapselter E-Mail, durch ein hybrides Kryptosystem umzusetzen, bei dem die Integrität der verbleibenden Teile weiterhin verifiziert werden kann. Ausgelagerte Teile sollen ebenfalls einzeln verifizierbar sein. Für die Umsetzung sind mehre Wege denkbar. Zwei Ansätze werden hier untersucht und gegenüber gestellt. Der erste Ansatz wäre die Verwendung von sogenannte Sanitizable Signatures. Ein weiterer Ansatz, der auch Rückwärtskompatibilität zu vorhandenen E-Mail-Lösungen verspricht, wäre das Hash-Tree-Verfahren.

> „Integrität bezeichnet die „Korrektheit (Unversehrtheit) von Daten und der korrekten Funktionsweise von Systemen “
>
> — Glossar des Bundesamtes für Sicherheit in der Informationstechnik

0.2 Grundlagen

0.2.1 Vorbereitung

Die Operation $s \xleftarrow{R} S$ beschreibt, dass ein Element s zufällig aus der Menge S gezogen wird. $c \leftarrow F(a, b\ldots)$ zeigt, dass ein Algorithmus F mit den Eingaben $(a, b\ldots)$ c als Ergebnis liefert. Die Notation $F(a; r)$ bezeichnet einen auf Zufall basierten Algorithmus F mit Eingabe a und Zufallsvariable r.

0.2.2 Aufbau einer Email

Der Aufbau einer E-Mail ist im RFC 5322 festgelegt. E-Mails sind intern in zwei Teile geteilt: Die Header-Felder mit Kopfzeilen und den Body (Textkörper) mit dem eigentlichen Inhalt der

Nachricht. Header-Felder sind Zeilen beginnend mit einem Feldnamen, gefolgt von einem Kolon („:“) und einem Feldkörper. Ein Header-Feld endet mit einem CRLF (Zeilenumbruch). Hier wird z.B. die Absenderangabe unter „From:“festgelegt. Innerhalb des Bodys werden weitere Untergliederungen definiert. Der Body einer E-Mail ist durch eine Leerzeile vom Header getrennt. Eine E-Mail darf gemäß RFC 5322 Abschnitt 2.3 nur Zeichen des 7-Bit-ASCII-Zeichensatzes enthalten. Sollen andere Zeichen, wie zum Beispiel deutsche Umlaute oder Daten wie zum Beispiel Bilder übertragen werden, müssen das Format im Header-Abschnitt deklariert und die Daten passend kodiert werden. Geregelt wird das durch RFC 2045. Außerdem müssen CR (Wagenrücklauf) und LF (Zeilenvorschub) zusammen als CRLF auftreten. Die Trennlinie der Untergliederung von Body wird in Header unter „Content-Type: “definiert.

0.2.3 Signatur-Schema

Ein Signatur-Schema kann man mit Hilfe von einem beliebigen IND-CPA-sicheren Verschlüsselungsverfahren via Black-Box-Reduktion konstruieren[Rom90]. wir betrachten folgendes Signatur-Schema SS = (Sigkg, Sign, Verify) . Der Verifikationsschlüssel vk kann mit Sigkg generiert werden. vk besteht aus einer Folge von Bits. Die Länge von vk interpretieren wir als Zahlen in $\{1, ..., 2^k\}$ mit Sicherheitsparameter k [Buc+11] [Lam79].

- Sigkg(1^k) erhält als Eingabe einen Sicherheitsparameter k in unärer Notation und gibt ein Schlüsselpaar (vk, sk) aus.
- Sign($1^k, sk, m$) erhält als Eingabe den Sicherheitsparameter 1^k, den Geheimschlüssel sk eine Nachricht m und gibt eine Signatur σ aus.
- Verify($1^k, vk, m, \sigma$) erhält als Eingabe den Sicherheitsparameter 1^k, den öffentlichen Schlüssel pk, eine Nachricht m, eine Signatur σ und gibt entweder 1 oder 0 aus.
  1 bedeutet, dass σ als Signatur für Nachricht m und öffentlichen Schlüssel pk akzeptiert wird. Bei 0 wird die Signatur nicht akzeptiert.

0.2.4 Commitment-Schema

Das Commitment-Schema ist ein grundlegender Bestandteil vieler kryptographischer Protokolle. Sie ermöglichen einer Partei, sich gegenüber einer anderen Partei auf einen Wert festzulegen, ohne etwas über diesen Wert zu verraten. Später kann dieser Wert dann aufgedeckt werden. Ein Commitment-Schema ein kryptographisches Zwei-Parteien- und Zwei-Phasen-Protokoll, welches aus Commit- und Reveal-Phase besteht: Com = (Commit, Reveal) [CLP10]. Der Sender bestimmt einen festen Wert und übergibt diesen dem Empfänger (Commit-Phase). Zum

Aufdecken (Reveal-Phase) bekommt der Empfänger von dem Sender den dazu nötigen Parameter. Damit das Verfahren korrekt ist, wird gefordert, dass der ursprüngliche Wert nach dem Aufdecken des Commitments wiederhergestellt sein muss. Die Algorithmen beider Parteien sollte man mit zwei PPT-Turingmaschinen umsetzen können. Dieses Protokoll muss außerdem folgende zwei Eigenschaften erfüllen:

Definition 0.1 (Binding) *Es darf nicht möglich sein, ein Commitment nachträglich auf einen anderen Wert zu ändern. Nachdem die Commit-Phase beendet ist, gibt es nur einen Wert, den ein möglicherweise schummelnder Sender den Empfänger offenlegen kann, sodass dieser akzeptiert [Ber13].*

Definition 0.2 (Hiding) *Das Commitment darf keinen Rückschluss auf den Wert zulassen, auf den sich der Committer festgelegt hat. Das muss auch gelten, falls der Empfänger zu schummeln versucht.*

1 Sanitizable Signatures

2 Aufbau von Hash Tree

2.1 Schema CommitVector

2.2 Schema HashTree

3 Realisierung in Java

3.1 Email Parser in Java

3.2 Sanitizable Signatures in Java

3.3 Schema HashTree in Java

4 Vergleich von Ergebnissen

5 Zusammenfassung und Ausblick