Signieren mit dem Full-Domain-Hash Schema

(orig. How to Sign with RSA and Rabin)

Ausarbeitung im Rahmen des Seminars zur Vorlesung

Kryptokomplexität I

von

Daniel Braune

bei

Prof. Dr. Jörg Rothe

Abstract

Diese Ausarbeitung befasst sich mit digitalen Signaturen, deren Funktionsweisen und Sicherheit, insbesondere wird das Full-Domain-Hash (FDH) Schema vorgestellt. Außerdem wird das Signaturverfahren RSA betrachtet, welches im Full-Domain-Hash Schema als Einwegpermutation dient. Desweiteren wird FDH auf seine Sicherheit untersucht und gezeigt, dass dieses $(t, \varepsilon) - \textit{secure}$ im Zufallsorakelmodell ist.

Inhaltsverzeichnis

Kapitel 1

Digitale Signaturen

Unterschriften sind im wahren Leben wichtig, um die Kommunikation einer Person eindeutig zuordnen zu können, auch wenn diese nicht persönlich bekannt ist. So sind beispielsweise Verträge oder Bestellungen ohne Signatur nicht möglich und dienen bei Problemen als Beweismittel, dass diese Kommunikation auch stattgefunden hat. Analog zur Unterschrift kann die digitale Signatur in einer elektronischen Kommunikation den Kommunikationspartner eindeutig identifizieren und so sicherstellen, dass mit genau diesem kommuniziert wird. Wird eine einfache E-Mail betrachtet, so kann der Empfänger zwar die Absenderadresse sehen, jedoch ist nicht sichergestellt, dass diese nicht gefälscht worden ist und die Nachricht auch von der Person stammt, die man erwartet hat. Um dieses Problem zu beheben, besteht die Möglichkeit, digitale Nachrichten zu signieren.

1.1 RSA Kryptosystem (Rivest, Shamir und Adleman)

Mit dem RSA-Verfahren können Nachrichten nicht nur verschlüsselt, sondern auch digital signiert werden. Dies funktioniert mit Hilfe eines privaten (private key) und einem öffentlichen (public key) Schlüssel. Der private Schlüssel ist hierbei nur dem Absender bekannt, der öffentlich Schlüssel ist frei zugänglich. Anders als beim normalen Verschlüsseln mit RSA, bei dem eine Nachricht mit dem öffentlichen Schlüssel des Empfängers verschlüsselt wird und dieser die Nachricht mit seinem privaten Schlüssel wieder entschlüsseln kann, wird bei der Signatur die gesendete Nachricht wird mit dem privaten Schlüssel vom Absender verschlüsselt und kann vom Empfänger nur mit dem öffentlichen Schlüssel des Absenders wieder entschlüsselt werden. Die Nachricht wurde daher mit dem privaten Schlüssel von dem Empfänger signiert und stammt daher eindeutig von ihm.

Definition: RSA

Sei (N,e) der öffentliche Schlüssel und (N,d) der Private, wobei N das Produkt aus zwei beliebigen $\frac{k}{2}$-bit Primzahlen p und q mit $2^{\frac{k}{2}-1} \leq p,q < 2^{\frac{k}{2}}$ ist, $e,d \in \mathbb{Z}^*_{\varphi(\mathbb{N})}$ mit $ed \equiv 1 \; mod \; \varphi(N)$ und $\varphi(N) = (p-1)(q-1)$.
Die RSA-Verschlüsselungsfunktion ist definiert als $f: \mathbb{Z}^*_N \to \mathbb{Z}^*_N$ mit $f(x) = x^e \; mod \; N$.
Die RSA-Entschlüsselungsfunktion ist definiert als $f^{-1}: \mathbb{Z}^*_N \to \mathbb{Z}^*_N$ mit $f^{-1}(y) = y^d \; mod \; N$
und $x,y \in \mathbb{Z}^*_N$. [BR96]

Beispiel: Signieren mit RSA

Um eine digitale Signatur mit RSA zu verdeutlichen, wird folgendes Beispiel betrachtet:

Sei $p = 17$ und $q = 5$.
Dann ist $N = p \cdot q = 17 \cdot 5 = 85$.
Nun kann $\varphi(N)$ berechnet werden mit $\varphi(N) = (17-1)(5-1) = 16 \cdot 4 = 64$.

Als nächstes wird eine Zahl e gewählt, die teilerfremd zu $\varphi(N) = 64$ ist. Sei also z.B. $e = 13$.

Aus der Definition 1.1 ist bekannt, dass $ed \equiv 1 \; mod \; \varphi(N)$ gilt. Daraus folgt für d

$$\begin{aligned} 13 \cdot d &\equiv 1 \; mod \; 64 \\ 13 \cdot d &= 1 + 64 \\ d &= 5 \end{aligned}$$

Betrachter wird nun die Nachricht m=C, die signiert werden soll. Hierbei wird C als Zahl dargestellt, also C $\Rightarrow$ 2.
Für die signierte Nachricht s wird $s = m^d mod \; N$ berechnet. Daraus folgt für die Signierung von $m = 2$:

$$\begin{aligned} s &= 2^5 \; mod \; 85 \\ &= 32 \; mod \; 85 \\ &= 32 \end{aligned}$$

Der signierte Nachticht zu $m = 2$ ist $s = 32$.

Zum Entschlüsseln wird vom Empfänger nun der öffentliche Schlüssel des Absenders gebracht, um

$m = s^e mod\ N$ zu berechnen.

$$\begin{aligned} m &= 32^{13}\ mod\ 85 \\ &= 36893488147419103232\ mod\ 85 \\ &= 2 \end{aligned}$$

Der versendete Nachricht ist $m = 2$, was wiederum dem zu verschlüsselten $m = C$ entspricht. Die Nachricht wurde somit eindeutig durch den Sender signiert und der Empfänger kann sich sicher sein, dass diese auch von dem zu erwartenden Absender stammt.

Kapitel 2

Full-Domain-Hash Schema (FDH)

Das Full-Domain-Hash Schema ist ein Signaturverfahren, welches Empfänger und Sender einer Nachricht ermöglicht, zu überprüfen, ob diese durch Dritte verändert wurde.

2.1 RSA Generator - Definition

Der RSA Generator, kurz $\mathscr{RSA}$, erstellt bei einem Input 1^k mit $k \in \mathbb{N}$ ein Paar aus $\frac{k}{2}$-bit Primzahlen p und q, welche multipliziert werden und somit das N mit $N = p \cdot q$ erstellt werden kann. Außerdem wird ein Verschlüsselungsexponent $e \in \mathbb{Z}^*_{\varphi(N)}$ und der zugehörige Entschlüsselungsexponent d gebildet, sodass $ed \equiv 1 \ mod \ \varphi(N)$ gilt. Der Generator gibt hierbei N, e, d mit $f: \mathbb{Z}^*_N \to \mathbb{Z}^*_N$ und $f^{-1}: \mathbb{Z}^*_N \to \mathbb{Z}^*_N$ zurück, welche wie folgt definiert sind: $f(x) = x^e \ mod \ N$ und $f^{-1} = y^d \ mod \ N$. [BR96]

2.2 FDH Schema

Das FDH-Verfahren ist ein asymmetrisches[1] Signaturverfahren, dessen Prinzip darin besteht, eine Nachricht zuerst zu hashen und anschließend eine beliebige Trapdoor-Einwegpermutation[2] darauf anzuwenden (in diesem Fall RSA). Das FDH Schema FDH=$(GenFDH, SignFDH, VerifyFDH)$ ist folgender Maßen definiert:

Der Key-Generator, welcher die Eingabe 1^k erhält, startet $\mathscr{RSA}(1^k)$, um (N, d, e) zu erhalten. Als Ausgabe werden privater (p_k) und öffentlicher Schlüssel (s_k) mit $p_k = (N, e)$ und $s_k = (N, d)$ zurück gegeben. Zum signieren und verifizieren wird außerdem eine Hashfunktion $H_{FDH}: \{0,1\}^* \to \mathbb{Z}^*_N$

[1] Die Asymmetrie bedeutet hierbei, dass aus dem Schlüssel d zum Verschlüsseln des jeweiligen Klartext kein Schlüssel e zur Entschlüsselung generiert werden kann. [Rot08]

[2] Funktionen, die sich nur umkehren lassen, wenn gewisse Zusatzinformationen vorhanden sind

($\mathbb{Z}_N^*$ wird als ideal angenommen) benötigt. Mit dieser Funktion wird der Klartext zuerst gehashed, bevor dieser Hashwert mit dem RSA-Verfahren verschlüsselt wird. Die Signierung und Verifizierung verlaufen wie folgt:

Signierung:

$$SignFDH_{N,d}(M)$$
$$y \leftarrow H_{FDH}(M)$$
$$\textbf{return } y^d \ mod \ N$$

Verifizierung:

$$VerifyFDH_{N,e}(M,x)$$
$$y \ \leftarrow \ x^e \ mod \ N \ ; \ y' \ \leftarrow \ H_{FDH}(M)$$
$$\textbf{if } y = y' \textbf{ then return 1 else return 0}$$

[BR96] Die Signierungsfunktion gibt hierbei $Sig(M)$ einer Nachricht M aus, welche dem Empfänger mit der signierten Nachricht übermittelt wird. Dieser kann anschließend mit der Verifizierungsfunktion überprüfen, ob die Nachricht tatsächlich von dem erwartenden Empfänger erstellt wurde. Hierzu gibt die Funktion 1 aus, wenn die Nachricht erfolgreich verifiziert wurde, 0 wenn nicht.

Beispiel: FDH

Um die Funktionsweise von FDH zu verdeutlichen, wird folgendes Beispiel betrachtet:

Gegeben ist eine Nachricht X, welche zum Text Y verschlüsselt werden soll. Auf diese Nachricht wird zuerst die Hashfunktion $H : \{0,1\}^* \rightarrow \mathbb{Z}_N^*$ angewendet, bevor anschließend die Signatur mit dem RSA Verfahren mit dem Schlüssel (N,e,d) durchgeführt wird. Der Schlüsseltext ergibt sich dann wie folgt:

$$X \rightarrow X_1 = H(X) \rightarrow X_2 = (X_1)^d \ mod \ N \rightarrow Y$$

2.3 Sicherheit von FDH

Um die Sicherheit von FDH zu betrachten, wird zuerst ein Zufallsorakelmodell eingeführt.

2.3.1 Das Zufallsorakelmodell

Da die exakten Eigenschaften von effizienten Hashfunktionen nur schwer erfassbar sind, wird eine einfache mathematische Abstraktion für Hashfunktionen benötigt. Ideal wäre hierfür eine Funktion $H(x)$, die für alle Inputs $X \in \{0,1\}^*$ einen Wert $Y = H(x)$ erstellt, indem Y zufällig aus $\{0,1\}^n$ mit $n \in \mathbb{N}$ gewählt wird. Da eine solche Funktion jedoch nicht implementiert werden kann, wird dieser Sachverhalt durch ein Zufallsorakel simuliert. Dieses Zufallsorakel besteht aus einer nicht leeren Liste von Paaren (X,Y), wobei X der Input ist. Falls kein Eintrag (X,Y) in der Liste vorhanden ist, soll ein zufälliges $Y \in \{0,1\}^n$ gewählt und das so entstandene Paar (X,Y) in die Liste eingetragen werden. Die Funktion H gibt zum Input X den Wert $H(X) = Y$ aus.

2.3.2 Definition: Inverting Algorithmus

Ein *Inverting Algorithmus* I für RSA bekommt als Eingabe (N,e,y) und versucht ein inverses $f^{-1}(y)$ zu finden. Die Erfolgswahrscheinlichkeit ist hierbei die Wahrscheinlichkeit, dass die Ausgabe dieses Inverse $f^{-1}(y)$ ist, wenn (N,e,d) bei der Durchführung von $\mathscr{RSA}(1^k)$ erhalten wird. $f(x)$ wird gleich y gesetzt um ein zufällig gewähltes $x \in \mathbb{Z}_N^*$ zu finden.
Ein Inverting Algorithmus ist ein t *Inverter* mit $t: N \to N$, wenn seine Laufzeit inklusive der Größe seiner Beschreibung an $t(k)$ gebunden ist. [BR96]

2.3.3 Definition: (t,ε)-breaks und (t,ε)-secure

Der Inverting Algorithmus I $(t,\varepsilon) - breaks\ \mathscr{RSA}$ mit $\varepsilon: N \to [0,1]$, wenn I ein t-Inverter und für jedes k die Erfolgswahrscheinlichkeit von I $\varepsilon(k)$ ist.
$\mathscr{RSA}$ ist (t,ε)-secure, wenn es keinen Inverter mit $(t,\varepsilon) - breaks\ \mathscr{RSA}$ gibt. [BR96]

2.3.4 Theorem: FDH ist $(t,\varepsilon) - secure$

Das folgende Theorem sowie der anschließende Beweis gelten für das Zufallsorakelmodell (siehe Kapitel 2.3.1). Angenommen $\mathscr{RSA}$ ist $(t',\varepsilon') - secure$. Sei q_{sig} die Anzahl an Orakelanfragen fürs Signieren und q_{hash} die Anzahl der Orakelanfragen fürs Hashen und ε die Erfolgswahrscheinlichkeit. Dann gilt für jedes q_{sig} und q_{hash}, dass FDH $(t,q_{sig},q_{hash},\varepsilon) - secure$ ist mit

$$t(k) = t'(k) - [q_{hash}(k) + q_{sig}(k) + 1] \cdot \Theta(k^3)$$
$$\varepsilon(k) = [q_{sig}(k) + q_{hash}(k) + 1] \cdot \varepsilon'(k)$$

$q_{hash} + q_{sig} + 1$ ist hierbei die Gesamtzeit der Orakelanfragen, $\Theta(k^3)$ die Zeit, die RSA benötigt.
[BR96] Wenn es einen Algorithmus gibt, der eine existenzielle Fälschung für das Full-Domain-Hash-Schema mit einer Laufzeit von t und Erfolgswahrscheinlichkeit von ε erstellen kann und dabei höchstens q_{hash} berechnet und höchstens q_{sig} Signaturen benötigt, dann gibt es einen Algorithmus, der diskrete Logarithmen von RSA-Modulen mit einer Laufzeit von t' und Erfolgswahrscheinlichkeit von ε' berechnet

2.3.5 Definition: $(t, q_{sig}, q_{hash}, \varepsilon)$-secure

Wenn $\mathscr{RSA}$ $(t'.\varepsilon')$-secure ist, dann ist ein Signaturschema $\Pi = (Gen, Sig, Verify)$ $(t, q_{sig}, q_{hash}, \varepsilon)$-secure. Der nachfolgende Beweis beinhaltet einen Fälscher F, welcher $(t, q_{sig}, q_{hash}, \varepsilon)$-breaks Π und erstellt aus F einen Inverter I, welcher (t', ε')-breaks $\mathscr{RSA}$. q_{sig} und q_{hash} sind dabei gegeben, t' soll so groß wie möglich sein, ε' so klein wie möglich. Dies ist erfüllt, wenn gilt: $t' = t - poly(q_{hash}, q_{sig}, k)$ und $\varepsilon' \approx \varepsilon$.

2.3.6 Beweis: FDH ist $(t, \varepsilon) - secure$

Sei F ein Fälscher mit $F = (t, q_{sig}, q_{hash}, \varepsilon) - breaks$ FDH und I ein Inverter, der $(t', \varepsilon') - breaks$ $\mathscr{RSA}$.

I bekommt als Input (N, e, y), wobei N, e und d beim Durchlaufen von $\mathscr{RSA}(1^k)$ erhalten werden und y zufällig aus $\mathbb{Z}_N^*$ gewählt wurde. Nun muss ein $x = f^{-1}(y)$ gefunden werden, wobei f die RSA Funktion mit N und e ist. Diese erstellt den öffentlichen Schlüssen (N, e) für FDH und startet F mit diesem Schlüssel. Der Fälscher F macht hierbei zwei verschiedene Zufallsabfragen: (1) Hash-Oracle-Abfrage und (2) Signing-Abfrage. Der Inverter I muss diese beiden Abfragen selbstständig beantworten. Einfachhalber wird angenommen, dass wenn F bereits eine Signing-Abfrage M durchgeführt hat, dass dann bereits ebenfalls eine Hash-Oracle-Abfrage durchgeführt wurde.
Sei nun $q = q_{sig} + q_{hash}$ und i ein ganzzahliger Counter, der anfänglich 0 ist. Der Inverter I wählt nun zufällig eine ganze Zahl j mit $j \in [1, ..., q]$.
Angenommen F bestimmt ein Hash-Oracle M, dann erhöht der Inverter I den Counter i und setzt $M_i = M$. Wenn $i = j$, dann wird $y_i = y$ gesetzt und y_i zurückgegeben. Andernfalls wird eine Zufallszahl r_i mit $r_i \in Z_N^*$ gewählt, $y_i = f(r_i)$ gesetzt und y_i zurück gegeben.
Anderseits, angenommen F macht eine Signing-Abfrage M. Nach der Annahme, dass dann bereits eine Hash-Abfrage von M gemacht wurde, ist $M = M_i$ für ein beliebiges i. I gibt dann die entsprechende Signatur r_i zurück.
Schließlich stoppt F und gibt als Ausgabe eine versuchte Fälschung (M, x) aus, wobei x die Ausgabe des Inverters I ist.
O.B.d.A wird angenommen, dass $M = M_i$ für ein beliebiges i. In diesem Fall, wenn (M, x) eine gültige

Fälschung ist, dann ist mit Wahrscheinlichkeit von mindestens $\frac{1}{q}$, mit $i = j$ und $x = f^{-1}(y_i) = f^{-1}(y)$ das korrekte Inverse von f gegeben.
Die Laufzeit von I ist hierbei die Summe aus der Laufzeit von F und der Zeit zur Bestimmung der y_i-Werte. Hauptsächlich fällt hierbei die RSA-Berechnung für jedes y_i ins Gewicht, welches in $O(n^3)$ oder besser berechnet werden kann.
Nun muss noch die Richtigkeit der Annahme gezeigt werden. Wenn F eine Signing-Abfrage macht, ohne vorher die entsprechende Hash-Abfrage zu machen, läuft I weiter und macht die Hash-Abfrage selbstständig, ähnlich auch für seine Ausgabe. Die Anzahl der Hash-Abfragen ist damit größtens $q_{hash} + q_{sig} + 1$, welches der Wert ist, der bereits oben benutzt wurde. □ [BR96]

Zusammenfassung

In dieser Ausarbeitung wurde das Full-Domain-Hash (FDH) Schema betrachtet mit RSA als Einwegpermutation. Um die Sicherheit von FDH zu überprüfen, wurde angenommen, dass FDH (t, ε) – *secure* ist, welches anschließend bewiesen wurde. Zusammenfassend kann gesagt werden, dass FDH im Zufallsorakelmodell ein sicheres Verfahren für die Signatur von Nachrichten ist, insbesondere bei chosen-message Angriffen. Generell ist FDH ein sicheres Verfahren zur Signierung von Nachrichten, jedoch gibt es weitere Verfahren, die sicherer sind.

Literaturverzeichnis

[BR96] BELLARE, Mihir; ROGAWAY, Philipp: The Exact Security of Digital Signatures - How to Sign with RSA and Rabin. In: *Advances in Cryptology - Eurocrypt 96 Proceedings* p.7-9 (1996).

[Rot08] ROTHE, Jörg: Komplexitätstheorie und Kryptologie. In: *Springer Verlag* (2008).