Institut für Informatik

Arbeitsgruppe Autonome Intelligente Systeme

Freiburg, 2. Februar 2016

Systeme I

Übungsblatt 12 - Speicherverwaltung und Sicherheit

Aufgabe 1 (1+2+2 Punkte)

In den bisherigen Übungsaufgaben haben Sie Seitentabellen kennengelernt. Nun betrachten wir die invertierte Seitentabelle. Die Hashfunktion sei gegeben durch

$$h(k_i) := k_i \bmod m,$$

wobei $m := 8$ die Anzahl der Rahmen des Hauptspeichers und k_i die Seitennummer ist. Abweichend zur Vorlesung soll der Wert der Hashfunktion direkt der Rahmennummer entsprechen und die Zeilen der invertierten Seitentabelle direkt den Rahmen des Hauptspeichers. Die invertierte Seitentabelle benötigt somit genau acht Einträge. Zusätzlich muss eine Liste der freien Rahmen verwaltet werden (welche gleichzeitig auch die freien Einträge in der invertierten Seitentabelle angibt). Es ergibt sich anfänglich folgende Tabelle und Liste:

Invertierte Seitentabelle:

Rahmennummer	Seitennummer	Überläufer
0		
1		
2		
3		
4		
5		
6		
7		

Liste der freien Rahmen:

0, 1, 2, 3, 4, 5, 6, 7

a) Ein Prozess reserviere die Seiten 0, 2, 1 und 6. Geben Sie die invertierte Seitentabelle und die Liste der freien Rahmen an.

b) Der Prozess reserviere weiterhin die Seiten 9 und 17. Rahmen für Überläufer sollen jeweils dem Anfang der Liste der freien Rahmen entnommen werden.

c) Auf wie viele Zeilen der Seitentabelle muss nun zugegriffen werden, um die Rahmennummer der Seite 17 zu finden? Welcher Nachteil ergibt sich daraus für invertierte Seitentabellen? Was ist hingegen der Vorteil?

Aufgabe 2 (2+3,5+1 Punkte)

In der Vorlesung wurde der *Secure Shell (SSH)*-Standard beschrieben, der einen sicheren Zugriff auf entfernte Rechner ermöglicht. In dieser Aufgabe betrachten wir den Schlüsselaustausch im Transport Layer mit der Diffie-Hellman-Methode[1].

Alice (Client) möchte eine SSH-Verbindung zu Bob (Server) aufbauen. Beide einigen sich auf die Verwendung des Diffie-Hellman-Schlüsselaustauschs und der oben dargestellten Blockziffer im Cipher-Block-Chaining-Mode (CBC).

Der Austausch besteht aus drei Schritten zwischen denen zwei Nachrichten ausgetauscht werden. Aus vorheriger Kommunikation sind die Identifikationsstrings ID_k und die ausgehandelten Parameter Init_k ($k \in \{\mathrm{Alice}, \mathrm{Bob}\}$) sowohl Alice als auch Bob bekannt.

Output : Nachricht m_1 an Bob
1 $a \leftarrow$ krypt. Zufallszahl mit $1 < a < p - 1$
2 $A \leftarrow g^a \bmod p$
3 $m_1 \leftarrow$ (`'SSH_MSG_KEXDH_INIT'`, A)

Algorithmus 1 : Berechnung von Alice

Nach Ausführen von Algorithmus 1 schickt Alice die Nachricht m_1 an Bob, dessen Berechnungen in Algorithmus 2 gelistet sind. Dabei bezeichnet $\mathcal{H}$ eine kryptographische Hashfunktion (wie z.B. SHA-1) und $\circ$ die Konkatenation („Aneinanderhängen“) von Strings.

Input : A aus m_1 von Alice
Output : Nachricht m_2 an Alice
1 $A \leftarrow$ Nachricht von Alice
2 $b \leftarrow$ krypt. Zufallszahl mit $1 < b < p - 1$
3 $B \leftarrow g^b \bmod p$
4 $K^{\mathrm{shared}} \leftarrow A^b \bmod p$
5 $h \leftarrow \mathcal{H}(\mathrm{ID}_{\mathrm{Alice}} \circ \mathrm{ID}_{\mathrm{Bob}} \circ \mathrm{Init}_{\mathrm{Alice}} \circ \mathrm{Init}_{\mathrm{Bob}} \circ K^{\mathrm{Bob}}_{\mathrm{öffentlich}} \circ A \circ B \circ K^{\mathrm{shared}})$
6 $s \leftarrow S(h, K^{\mathrm{Bob}}_{\mathrm{privat}})$ `/* Signatur, z.B. mit DSA */`
7 $m_2 \leftarrow$ (`'SSH_MSG_KEXDH_REPLY'`, $K^{Bob}_{\mathrm{öffentlich}}, B, s$)

Algorithmus 2 : Berechnung von Bob

Nun hat Bob bereits den gemeinsamen Schlüssel K^{shared} berechnet. Er schickt die Nachricht m_2 an Alice, so dass diese in Algorithmus 3 seine Identität verifizieren und ebenfalls K^{shared} berechnen kann.

Für die symmetrische Verschlüsselung verwenden wir in dieser Aufgabe der Einfachheit halber die Blockchiffre $(E, D) : \{0,1\}^8 \times \{0,1\}^4 \to \{0,1\}^4$, deren Ausgabe $C = E(K, P)$ für die zwei festen Schlüssel $K_0 = 2_{10} = \texttt{0000 0010}_2$ und $K_1 = 7_{10} = \texttt{0000 0111}_2$ wie folgt gegeben ist.

P	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
$E(K_0, P)$	0000	0001	1001	1110	1111	1011	0111	0110	1101	0010	1100	0101	1010	0100	0011	1000
$E(K_1, P)$	0010	0000	1001	1110	1111	1011	0111	0110	1101	0010	1100	0101	1010	0100	0011	1000

[1] spezifiziert in RFC 4253, `http://www.ietf.org/rfc/rfc4253.txt`

Input : $K^{Bob}_{\text{öffentlich}}$, B und s aus m_2 von Bob
1 **if** $K^{Bob}_{\text{öffentlich}} \neq$ *gespeicherter Schlüssel von Bob in* `~/.ssh/known_hosts` **then**
2 | breche mit Fehlermeldung ab
3 **end**
4 $K^{\text{shared}} \leftarrow B^a \bmod p$
5 $h \leftarrow \mathcal{H}(\text{ID}_{\text{Alice}} \circ \text{ID}_{\text{Bob}} \circ \text{Init}_{\text{Alice}} \circ \text{Init}_{\text{Bob}} \circ K^{\text{Bob}}_{\text{öffentlich}} \circ A \circ B \circ K^{\text{shared}})$

6 $v \leftarrow V(h, s, K^{\text{Bob}}_{\text{öffentlich}})$ `/* verifiziere Signatur */`
7 **if** $v \neq 1$ **then** `/* ungültige Signatur */`
8 | breche mit Fehlermeldung ab
9 **end**

Algorithmus 3 : Berechnung von Alice

a) Wir verwenden für den DH-Schlüsselaustausch die Parameter $p = 23$ und $g = 11$. Bob besitzt den öffentlichen Schlüssel $K^{\text{Bob}}_{\text{öffentlich}} = (23, 7, 17)$.

1) Betrachten Sie den Verlauf des Schlüsselaustauschs zwischen Alice und Bob. Berechnen Sie dabei die Werte für A, B sowie K^{shared} und geben Sie die Nachrichten m_1 und m_2 an. Gehen Sie davon aus, dass Alice die Zufallszahl 4 und Bob die Zufallszahl 19 zieht. Sie müssen keine Werte für h, s und V angeben.

Hinweis Wir empfehlen, für die Berechnung von $x^y \bmod z$ z.B. WolframAlpha (Eingabe `x^y mod z`) oder Python (`(x**y) % z`) zu verwenden, da mit anderen Programmen unter Umständen inkorrekte Ergebnisse durch numerische Ungenauigkeiten auftreten.

2) Nach dem erfolgreichen Schlüsselaustausch möchte Bob den Klartext $P =$ `0100 0111 0100 0001` an Alice schicken. Geben Sie das zugehörige Chiffrat C an, indem Sie die obige Blockchiffre mit dem gemeinsamen Schlüssel K aus dem vorherigen Aufgabenteil verwenden. Verwenden Sie $IV =$ `0000` als zufälligen Initialisierungsvektor.

Hinweis Im CBC-Mode wird jeder Klartextblock P_i mittels der XOR-Operation ($\oplus$) mit dem letzten Chiffratblock C_{i-1} (bzw. beim ersten Schritt dem Initialisierungsvektor) verknüpft, bevor er mit der Blockchiffre abgebildet wird.

b) Wir betrachten nun die Server-Authentifizierung durch das Signieren während des Schlüsselaustauschs.

1) Begründen Sie, warum die Authentifizierung des Servers (Bob) gegenüber dem Client (Alice) bereits während des Schlüsselaustauschs und nicht (wie die Authentifizierung des Clients) über eine gesicherte Verbindung nach Etablierung eines sicheren Kanals stattfindet.

2) Begründen Sie, warum sowohl die Bedingung in Zeile 1 als auch die Bedingung in Zeile 7 in Algorithmus 3 für die Sicherheit des Verfahrens nötig ist. Welches Problem ergibt sich in der Praxis bei der Überprüfung der ersten Bedingung?

3) Wird die Sicherheit des Verfahrens beeinträchtigt, wenn in Zeile 5 in Algorithmus 2 stattdessen $h \leftarrow \mathcal{H}(\mathrm{ID}_{\mathrm{Bob}})$ gesetzt wird?

c) Angenommen, Eve hat Zugriff auf einen Quantencomputer, der das Problem des diskreten Logarithmus effizient löst (z.B. mit Shor's Algorithm[2]). Ist der durch das SSH-Protokoll aufgebaute Kanal dann noch vertraulich? Begründen Sie bzw. geben Sie einen Angriff an.

Hinweis Für die Lösung dieses Aufgabenteils reicht die Annahme, dass das Problem des diskreten Logarithmus effizient lösbar ist; die Art der Lösung muss nicht betrachtet werden.

Abgabe: Als PDF-Datei im Ilias bis zum 08. Februar 2016, 23:59 Uhr

[2]als Exkurs (nicht relevant für die Vorlesung): Shor, P. 1999. "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer." SIAM Review 41 (2): 303–32.http://dx.doi.org/10.1137/S0036144598347011