Institut für Theoretische Informatik
Jun.-Prof. Dr. D. Hofheinz

Stammvorlesung Sicherheit im Sommersemester 2014

Klausur

22.07.2014

Vorname: _____

Nachname: _____

Matrikelnummer: _____

Hinweise

- Für die Bearbeitung stehen Ihnen 60 Minuten zur Verfügung.
- Zum Bestehen der Klausur sind 20 der 60 möglichen Punkte hinreichend.
- Es sind keine Hilfsmittel zugelassen.
- Schreiben Sie Ihre Lösungen auf die Aufgabenblätter sowie auf deren Rückseiten.
- Zusätzliches Papier erhalten Sie bei Bedarf von der Aufsicht.

Aufgabe	mögliche Punkte					erreichte Punkte				
	a	b	c	d	Σ	a	b	c	d	Σ
1	2	3	3	3	11					
2	4	6	-	-	10			-	-	
3	3	4	4	-	11				-	
4	3	4	4	-	11				-	
5	7				7					
6	10x1				10					
Σ					60					

Aufgabe 1. (2+3+3+3 Punkte) Betrachten Sie das einfache Lehrbuch-RSA-Verschlüsselungsverfahren aus der Vorlesung.

(a) Seien die Primzahlen $P = 7$ und $Q = 17$ gegeben. Berechnen Sie den geheimen Exponenten d zum öffentlichen Exponenten $e = 5$ und geben Sie den öffentlichen Schlüssel an.

(b) (i) Verschlüsseln Sie die Nachricht $M = 97$ mit den Schlüsseln aus (a).

(ii) Entschlüsseln Sie das Chiffrat $C = 32$ mit den Schlüsseln aus (a).

Hinweis: Rechnen Sie vorteilhaft! Zerlegen Sie die Zahlen geschickt in einzelne Faktoren und verwenden Sie bei Bedarf negative Repräsentanten modulo N.

(c) Das betrachtete Lehrbuch-RSA-Verfahren ist IND-CPA-unsicher.

(i) Geben Sie einen effizienten Angreifer an, der das IND-CPA-Experiment mit Wahrscheinlichkeit 1 gewinnt.

(ii) Nennen Sie eine Möglichkeit, wie man diesen Angriff verhindern könnte.

(d) Wir betrachten folgendes Sicherheitsspiel für das Lehrbuch-RSA-Verfahren: Sei $pk = (N, e)$ mit $N = PQ$ bekannt. Ein Angreifer bekommt vom Challenger ein Chiffrat C^* zur Nachricht M^* und soll M^* zurückgeben. Er darf sich zusätzlich Chiffrate $C \neq C^*$ vom Challenger entschlüsseln lassen. Geben Sie einen effizienten Angreifer mit Erfolgswahrscheinlichkeit 1 an.

Lösungsvorschlag zu Aufgabe 1.

(a) Wir berechnen $N := PQ = 7 \cdot 17 = 119$ und $\varphi(N) = (P-1)(Q-1) = 6 \cdot 16 = 96$. Der erweiterte euklidische Algorithmus $\mathsf{EE}(5, 96)$ liefert $(-19, 1)$, da

$$\begin{aligned} & 96 = 5 \cdot 19 + 1 \\ \Leftrightarrow \; & -19 \cdot 5 + 96 = 1 \end{aligned}$$

gilt. Der geheime Exponent lautet $d := -19 \bmod 96 = 77$. ($d = -19$ ist ebenso eine gültige Lösung.) Der öffentliche Schlüssel ergibt sich zu $(N, e) = (119, 5)$.

(b) (i) Wir berechnen

$$\begin{aligned} C := M^e \bmod N &= 97^5 \bmod 119 = (-22)^5 \bmod 119 = (-11 \cdot 2)^5 \bmod 119 \\ &= (-11)^2(-11)^2(-11) \cdot 2^5 \bmod 119 = 2 \cdot 2 \cdot 2^5 \cdot (-11) \bmod 119 \\ &= 2^7 \cdot (-11) \bmod 119 = 128 \cdot (-11) \bmod 119 = -99 = 20 \bmod 119. \end{aligned}$$

(Dabei ist $(-11)^2 = 121 \bmod 119 = 2 \bmod 119$.)

(ii) Wir berechnen $C^d = M \bmod N$

$$C^{77} \bmod 119 = (32)^{77} \bmod 119 = (2^5)^{77} \bmod 119 = 2^{1 \bmod 96} \bmod 119 = 2 \bmod 119$$

da $d = 77$ das Inverse von $e = 5 \bmod \varphi(119)$, d.h. $e \cdot d = 1 \bmod \varphi(N)$, ist.

(c) (i) Ein Angreifer erhält vom Experiment den öffentlichen Schlüssel (N, e) und wählt sich zwei Nachrichten M_0, M_1, wobei $|M_0| = |M_1|$. Diese schickt er an das Experiment und bekommt das Chiffrat C_b, $b \xleftarrow{\$} \{0, 1\}$ von M_b zurück und soll b richtig bestimmen. Da die Verschlüsselung in diesem Falle deterministisch ist, kann er einfach testen, ob $M_0^e \stackrel{?}{=} C_b$. Falls ja, gibt er $b = 0$ zurück, ansonsten $b = 1$ und gewinnt mit Wahrscheinlichkeit 1.

(ii) Die Verschlüsselungsfunktion muss randomisiert werden, z.B. durch zufälliges Padding.

(d) Folgender erfolgreiche Angriff ist möglich, der die Homomorphie des Verfahrens ausnutzt:

1. Der Angreifer wählt zuerst eine zufällige Nachricht $R \xleftarrow{\$} \mathbb{Z}_N \setminus \{1\}$.
2. Er berechnet $C \cdot R^e = (M^*)^e \cdot R^e = (M^* \cdot R)^e = C'$.
3. Er lässt sich C' vom Challenger entschlüsseln und erhält somit $M^* \cdot R$.
4. Nun kann er das Inverse von R, $R^{-1} \bmod N$ berechnen, da er R und N kennt und somit $M^* \cdot R \cdot R^{-1} = M^*$

Aufgabe 2. (4+6 Punkte)

(a) Gegeben sei ein MAC für zwei Parteien A und B:

(i) Beschreiben Sie allgemein die Algorithmen eines MAC-Verfahrens und eventuelle Annahmen an die Parteien bzw. Anforderungen an die Algorithmen.

(ii) Nennen Sie 2 Ziele, die durch ein MAC-Verfahren sichergestellt werden sollen.

(iii) Was ist der Unterschied zu einer digitalen Signatur?

(b) Betrachten wir nun das ElGamal-Signaturverfahren $(\mathsf{Gen}, \mathsf{Sig}, \mathsf{Ver})$ über der zyklischen Gruppe $\mathbb{G} = \langle g \rangle$.

(i) Sei $h \in \mathbb{G}$. Geben Sie den Secret Key sk zum Public Key $pk = (\mathbb{G}, g, h)$ an.

(ii) Geben Sie einen Angriff an, der zeigt, dass das Verfahren aus der Vorlesung nicht EUF-CMA-sicher ist.

(iii) Wie kann man Ihren Angriff aus (ii) verhindern?

(iv) Wir verändern den Signaturalgorithmus zu $\mathsf{Sig}(sk, M) = \sigma$, wobei $\sigma \cdot x = M \bmod |\mathbb{G}|$ für das geheime x aus dem Secret Key gilt. Wie sieht die entsprechende Verifikation aus und wieso ist diese Änderung problematisch?

Lösungsvorschlag zu Aufgabe 2.

(a) (i) Annahme: Beide Parteien besitzen gemeinsames Geheimnis/Schlüssel K.
Algorithmen:
- $\sigma \leftarrow \mathsf{Sig}(K, M)$ signiert mit K die Nachricht M und liefert σ.
- $\mathsf{Ver}(K, M, \sigma) \in \{0, 1\}$ überprüft mit K, ob σ für die Nachricht M gültig $= 1$ oder ungültig $= 0$ ist.

Korrektheit: $\mathsf{Ver}(K, M, \sigma) = 1$ für alle K, M und $\sigma \leftarrow \mathsf{Sig}(K, M)$.

(ii)
- Authentifikation
- Integrität der Nachricht

(iii) Bei einer Digitalen Signatur gibt es einen geheimen Schlüssel sk zum Signieren und einen öffentlichen Schlüssel pk zum Verifizieren. Diese werden durch einen Schlüsselerzeugungsalgorithmus $(pk, sk) \leftarrow \mathsf{Gen}(1^k)$ erzeugt.

(b) (i) $sk = (\mathbb{G}, g, \log_g h)$

(ii)
- 1.Möglichkeit: Der Angreifer $\mathcal{A}$ wählt $\sigma = (a, b) := (g^x, -g^x)$ (mit g^x aus dem öffentlichen Schlüssel), welches eine gültige Signatur für $M = 0$ ist.
- 2.Möglichkeit: Der Angreifer $\mathcal{A}$ wählt z zufällig und berechnet $a := g^z g^x = g^{z+x}$. Damit ist $\sigma = (a, -a)$ eine gültige Signatur für die Nachricht $M = -az$.

(iii) Hash-Then-Sign Paradigma: Man verwendet eine kollisionsresistente Hashfunktion H und signiert nicht M sondern $H(M)$.

(iv) $\mathsf{Ver}(pk, M, \sigma)$: mit $pk = g^x$ überprüfe, ob $(g^x)^\sigma = g^M \bmod |\mathbb{G}|$.
Problem: Der geheime Exponent x kann bei diesem Verfahren berechnet werden: $\sigma \in \mathbb{G}$, die Nachricht M und die Gruppenordnung $|\mathbb{G}|$ sind jeweils bekannt und somit kann mit dem erweiterten euklidischen Algorithmus $\sigma^{-1} \bmod |\mathbb{G}|$ berechnet werden und dadurch $x = M \cdot \sigma^{-1} \bmod |\mathbb{G}|$.

Aufgabe 3. (3+4+4 Punkte)

(a) Beschreiben Sie für eine Blockchiffre $E(K, X) : \{0,1\}^k \times \{0,1\}^{2k} \to \{0,1\}^{2k}$, wie sie eine Nachricht $M \in \{0,1\}^*$ im CBC-Mode ver- und entschlüsselt und nennen Sie einen Vorteil gegenüber dem ECB-Mode.

(b) Aus der Vorlesung ist die Blockchiffre DES bekannt. Hierbei handelt es sich um eine Feistel-Chiffre mit einer Blocklänge von 64 Bit und einer effektiven Schlüssellänge von 56 Bit. Wir betrachten eine Variante von DES, bei der die Rundenschlüssel nicht aus dem Gesamtvorrat von 56 Bit berechnet werden, sondern in den Feistel-Runden 1-8 mittels einer beliebigen Funktion aus der ersten 28-Bit-Schlüsselhälfte erzeugt werden und in den Feistel-Runden 9-16 mittels derselben Funktion aus der zweiten 28-Bit-Schlüsselhälfte erzeugt werden.

Konstruieren Sie einen Angriff, mit dem sich der zur Verschlüsselung benutzte geheime Schlüssel K bestimmen lässt, und der höchstens 2^{40} Operationen benötigt. Ihr Angriffsalgorithmus ist in Besitz von einigen Klartext-Chiffrat-Paaren. Erläutern Sie kurz, warum Ihr Angriff tatsächlich mit hoher Wahrscheinlichkeit den richtigen Schlüssel findet und zeigen Sie kurz, dass Ihr Angriff tatsächlich höchstens 2^{40} Operationen benötigt.

Hinweis: Angriffe, die weniger als 2^{40} Operationen benötigen sind ebenfalls als Lösung zulässig. Nehmen Sie an, dass ein Algorithmus zur Sortierung von n Elementen exakt $n \cdot \log n$ Operationen benötigt.

(c) Sei $n \in \mathbb{N}, n \geq 2$. Wir betrachten eine symmetrische Chiffre

$$\begin{aligned} \text{Enc} : \{0,1\}^{2n} \times \{0,1\}^{2n} &\to \{0,1\}^{2n}, \\ \text{Enc}(K, M) &= C = C_1 \dots C_{2n}, \end{aligned}$$

die Nachrichten $M = M_1 \dots M_{2n} \in \{0,1\}^{2n}$ mit einem $2n$-Bit-Schlüssel $K = K_1 \dots K_{2n}$ folgendermaßen verschlüsselt:

M_1	M_2	$\dots$	M_n	M_{n+1}	M_{n+2}	$\dots$	M_{2n}
$\oplus$	$\oplus$	$\dots$	$\oplus$	$\oplus$	$\oplus$	$\dots$	$\oplus$
K_1	K_3	$\dots$	K_{2n-1}	K_{2n}	K_{2n-1}		K_{n+1}
$\oplus$	$\oplus$	$\dots$	$\oplus$	$\oplus$	$\oplus$	$\dots$	$\oplus$
K_2	K_4	$\dots$	K_{2n}	K_1	K_2		K_n
C_1	C_2	$\dots$	C_n	C_{n+1}	C_{n+2}	$\dots$	C_{2n}

Wie kann ein Angreifer geschickt zwei Nachrichten $M^{(1)} \neq M^{(2)}$ gleicher Länge wählen und **nur** mithilfe des Chiffrats und der gewählten Nachrichten entscheiden, welche der beiden verschlüsselt wurde?

Lösungsvorschlag zu Aufgabe 3.

(a) Verschlüsselung:

- Teile M in $2k$-Bit Blöcke auf. $M_1, \dots \in \{0,1\}^{2k}$ (fülle evtl. mit einem Padding auf)
- Setze $C_0 := IV$ (zufälliger Initialisierungsvektor)
- $C_i := E(K, M_i \oplus C_{i-1})$

Entschlüsselung: $M_i = D(K, C_i) \oplus C_{i-1}$
Vorteil: Im Vergleich zum ECB-Mode werden gleiche Nachrichtenblöcke nicht zu gleichen Chiffratblöcke verschlüsselt.

(b) Wir verwenden ein Vorgehen wie bei dem aus der Vorlesung bekannten Meet-In-The-Middle-Angriff auf 2DES. Es bezeichne

$$\text{DES}_1 : \{0,1\}^{28} \times \{0,1\}^{64}$$

die Berechnung der ersten acht DES-Runden mit einem 28-Bit Teilschlüssel und einer 64-Bit Nachricht. Entsprechend bezeichne

$$\mathrm{DES}_2^{-1} : \{0,1\}^{28} \times \{0,1\}^{64}$$

die Berechnung der letzten acht DES-Runden bei der Entschlüsselung mit einem 28-Bit Teilschlüssel und einem 64-Bit Chiffrat. Der genaue Angriffsalgorithmus ist:

1. Wähle ein Nachricht-Chiffrat-Paar $M^*, C^* \in \{0,1\}^{64}$.
2. Berechne eine Liste L der Tupel $(K_1, \mathrm{DES}_1(K_1, M^*))$ für alle $K_1 \in \{0,1\}^{28}$.
3. Sortiere L nach der zweiten Komponente.
4. Für alle Schlüsselhälften $K_2 \in \{0,1\}^{28}$:
   (i) Berechne das Zwischenergebnis $D := \mathrm{DES}_2^{-1}(K_2, C^*)$ der Entschlüsselung mit dem DES-Algorithmus nach 8 Runden mit Schlüsselhälfte K_2 und Chiffrat C^*.
   (ii) Falls es einen Eintrag (K_1, D) in der Liste L gibt (binäre Suche), dann ist (K_1, K_2) ein Schlüsselkandidiat.
5. Überprüfe Schlüsselkandidaten durch weitere Nachrichten-Chiffrat-Paare M und C.

Es ist $C^* = \mathrm{DES}(K, M^*) = \mathrm{DES}_2(K_2, \mathrm{DES}_1(K_1, M^*))$. Deshalb gilt $\mathrm{DES}_2^{-1}(K_2, C^*) = \mathrm{DES}_1(K_1, M^*)$. Da man über alle Schlüsselhälften K_1, K_2 iteriert, wird der echte Schlüssel unter den Schlüsselkandidaten sein.

Es werden nur wenige Schlüsselkandidaten erwartet, diese kann man wie in Schritt 5. überprüfen.

Insgesamt benötigt der Algorithmus etwa $2 \cdot 2^{28} = 2^{29}$ Aufrufe der 8-Runden-DES-Operation. Hinzu kommt das Sortieren der Liste L, das $28 \cdot 2^{28}$ Schritte benötigt, und 2^{28} binäre Suchen, die jeweils etwa 28 Operationen benötigen. Insgesamt ergibt sich also ein Aufwand von etwa $2^{28} + 2^{28} + 28 \cdot 2^{28} + 28 \cdot 2^{28} \leq 64 \cdot 2^{28} = 2^{34}$.

(c) 1: Wir wählen $M^{(1)} = 0^{2n}$ und $M^{(2)}$, so dass $M_i^{(2)} = M_i^{(1)} = 0$ für $i \in \{1, \ldots, 2n-1\}$ aber $M_{2n}^{(2)} = 1$. Wir erhalten das Chiffrat $C^{(b)}$ (eine Verschlüsselung von $M^{(b)}$ für zufälliges $b \in \{1,2\}$). Wir berechnen

$$C_1^{(b)} \oplus \cdots \oplus C_{2n}^{(b)} = M_1^{(b)} \oplus \cdots \oplus M_{2n}^{(b)} = M_{2n}^{(b)}$$

Falls $M_{2n}^{(b)} = 0$ wurde $M^{(1)}$ verschlüsselt, falls $M_{2n}^{(b)} = 1$ wurde $M^{(2)}$ verschlüsselt.

2: Wir wählen $M^{(1)}$ und $M^{(2)}$ so, dass $M_i^{(1)} = M_i^{(2)} = 0$ für $i \in \{1, \ldots, 2n\} \setminus \{n\}$ und $M_n^{(1)} = 0 \neq M_n^{(2)} = 1$. Wir erhalten das Chiffrat $C^{(b)}$ (eine Verschlüsselung von $M^{(b)}$ für zufälliges $b \in \{1,2\}$). In beiden Fällen gilt $C_i^{(1)} = C_i^{(2)}$ für $i \in \{1, \ldots, 2n\} \setminus \{n\}$ und somit ist

$$\begin{aligned} C_1^{(b)} \oplus C_{n+1}^{(b)} \oplus C_{n+2}^{(b)} &= M_1^{(b)} \oplus K_1 \oplus K_2 \oplus M_{n+1}^{(b)} \oplus K_{2n} \oplus K_1 \oplus M_{n+2}^{(b)} \oplus K_{2n-1} \oplus K_2 \\ &= K_1 \oplus K_2 \oplus K_{2n} \oplus K_1 \oplus K_{2n-1} \oplus K_2 \\ &= K_{2n} \oplus K_{2n-1} \end{aligned}$$

Damit können wir nun $C_n^{(b)} \oplus K_{2n} \oplus K_{2n-1} = M_n^{(b)}$ berechnen und entscheiden, ob $b = 0$ oder $b = 1$ ist.

(Hinweis: Dies sind zwei Möglichkeiten die Aufgabe zu lösen, evtl. gibt es noch andere Möglichkeiten.)

Aufgabe 4. (3+4+4 Punkte) Eine vertrauenswürdige Instanz veröffentlicht einen RSA-Modulus $N = PQ$. Sie zieht zufällig gleichverteilt geheime Zahlen $s_1, \ldots, s_k \leftarrow \mathbb{Z}_n$ und veröffentlicht $v_i = s_i^2 \bmod N$ für alle $i \in \{1, \ldots, k\}$. Ein Prover P erhält die geheimen Zahlen s_i für alle $i \in \{1, \ldots, k\}$ und will einen Verifier V überzeugen, dass er die Quadratwurzeln von v_i kennt. Dazu wird folgendes Protokoll ausgeführt:

1. P wählt $r \xleftarrow{\$} \mathbb{Z}_N$, berechnet $x = r^2 \bmod N$, sendet x an V.
2. V wählt zufällig gleichverteilt $b_1, \ldots, b_k \leftarrow \{0, 1\}$, sendet b_i, für alle $i \in \{1, \ldots, k\}$ an P.
3. P berechnet $y = r \prod_{i=1}^{k} s_i^{b_i} \bmod N$, sendet y an V.
4. V überprüft, ob $y^2 = x \prod_{i=1}^{k} v_i^{b_i} \bmod N$ gilt.

Dies wird t Mal wiederholt.

(a) Zeigen Sie die Korrektheit des Protokolls für ehrlichen P und V.

(b) Zeigen Sie durch Angabe eines Simulators, dass die Zero-Knowledge-Eigenschaft gilt.

(c) Betrachten wir für $k = 1, t = 2$ den Spezialfall, dass jedes Mal der gleiche Zufall r verwendet wird. Wie muss V dabei vorgehen um geheime Informationen zu bekommen und welche sind dies?

Lösungsvorschlag zu Aufgabe 4.

(a) Wenn P die s_i für $i \in [k]$ kennt, wird er V davon überzeugen können, da

$$y^2 = (r \prod_{i=1}^{k} s_i^{b_i})^2 = r^2 \prod_{i=1}^{k} s_i^{2b_i} = x \prod_{i=1}^{k} v_i^{b_i} \bmod N$$

(b) Ein Simulator S, der die s_i nicht kennt, kann gültige Protokolltransskripte für alle möglichen V simulieren, indem er folgendes ausführt:

1. S wählt zufällige Bits b_i', zufälliges, gleichverteiltes $r \leftarrow \mathbb{Z}_N$ und berechnet $x = r^2 \prod_{i=1}^{k} v_i^{-b_i'} \bmod N$. Er sendet x an V.
2. V sendet zufällige Bits b_i an S.
3. Falls $b_i' = b_i$ für alle $i \in \{1, \ldots, k\}$ sendet S $y = r$, ansonsten löscht er das Transkript und startet die Simulation von vorne.

(c) V wählt folgende Taktik:

1. Im ersten Durchlauf sendet er $b = 0$ und erhält somit $y_1 = r \bmod N$.
2. Im zweiten Durchlauf wählt er $b = 1$ und erhält $y_2 = r \cdot s \bmod N$.
3. Nun kann er mit $y_2/y_1 = \frac{r \cdot s}{r} = s \bmod N$ das Geheimnis s berechnen.

Aufgabe 5. (7 Punkte) Im Bell-LaPadula-Modell aus der Vorlesung seien

- die Subjektmenge $\mathcal{S} = \{s_1, s_2, s_3\}$,
- die Objektmenge $\mathcal{O} = \{o_1, o_2, o_3, o_4\}$,
- die Menge der Zugriffsoperationen $\mathcal{A} = \{\texttt{read}, \texttt{write}, \texttt{append}, \texttt{execute}\}$ und
- die Menge der Sicherheitslevel $\mathcal{L} = \{\texttt{topsecret}, \texttt{secret}, \texttt{unclassified}\}$ mit der $\mathcal{L}$-Halbordnung $\texttt{topsecret} \geq \texttt{secret} \geq \texttt{unclassified}$

gegeben. Die Zugriffskontrollmatrix $M = (M_{s,o})_{s \in \mathcal{S}, o \in \mathcal{O}}$ ist durch die Tabelle

	o_1	o_2	o_3	o_4
s_1	{read, append, execute}	{read, write, append}	{read, append}	$\mathcal{A}$
s_2	{read}	{read, execute}	{read, append}	$\mathcal{A}$
s_3	$\emptyset$	{read}	{read, append}	{read, write, append}

definiert und die Zuordung der maximalen und aktuellen Sicherheitslevel $F = (f_s, f_c, f_o)$ ist durch die Tabellen

	$f_s(\cdot)$	$f_c(\cdot)$
s_1	topsecret	unclassified
s_2	secret	unclassified
s_3	unclassified	unclassified

	$f_o(\cdot)$
o_1	topsecret
o_2	topsecret
o_3	secret
o_4	unclassified

beschrieben. Betrachten Sie die folgende Abfolge von Zugriffen $b \in \mathcal{S} \times \mathcal{O} \times \mathcal{A}$ in Reihenfolge:

1. $(s_1, o_2, \texttt{read})$
2. $(s_2, o_2, \texttt{read})$
3. $(s_3, o_4, \texttt{execute})$
4. $(s_3, o_3, \texttt{append})$
5. $(s_2, o_3, \texttt{write})$
6. $(s_1, o_3, \texttt{write})$

Beschreiben Sie – beispielsweise in der Spalte "Bemerkungen" in der unter stehenden Tabelle, die Sie für Ihre Lösung benutzen können –, ob die einzelnen Zugriffe gültig (✓) oder ungültig (×) sind, und ob der aktuelle Sicherheitslevel nach gültigem Zugriff geändert wird. Falls der Zugriff nicht gültig ist, zeigen Sie auf, welche Eigenschaft(en) – im Sinne der ds-, ss- oder $\star$-Eigenschaft – verletzt wurde(n). Begründen Sie Ihre Entscheidung (ebenfalls beispielsweise in der Spalte "Bemerkungen" unten). Gehen Sie davon aus, dass zu Beginn noch kein Zugriff stattgefunden hat.

Zugriff	ds	ss	$\star$	Bemerkungen
1. $(s_1, o_2, \texttt{read})$				
2. $(s_2, o_2, \texttt{read})$				
3. $(s_3, o_4, \texttt{execute})$				
4. $(s_3, o_3, \texttt{append})$				
5. $(s_2, o_3, \texttt{write})$				
6. $(s_1, o_3, \texttt{write})$				

Lösungsvorschlag zu Aufgabe 5. Gültig sind die einzelnen Zugriffe $b \in \mathcal{S} \times \mathcal{O} \times \mathcal{A}$, falls die ds-, die ss- und die $\star$-Eigenschaft erfüllt sind und damit die Systemsicherheit erhalten bleibt. Erfüllt sind alle genannten Eigenschaften, falls in den Spalten „ds“, „ss“ und „$\star$“ das Symbol ✓ zu finden ist. In der unten stehenden Tabelle findet sich ein Lösungsvorschlag.

	Zugriff	ds	ss	$\star$	Bemerkungen
1.	$(s_1, o_2, \texttt{read})$	✓	✓	✓	aktueller Sicherheitslevel auf $f_c(s_1) = \texttt{topsecret}$ gesetzt
2.	$(s_2, o_2, \texttt{read})$	✓	×	✓	ss-Eigenschaft verletzt, da $f_s(s_2) < f_o(o_2)$
3.	$(s_3, o_4, \texttt{execute})$	×	✓	✓	ds-Eigenschaft verletzt, da $\texttt{execute} \notin M_{s_3,o_4}$
4.	$(s_3, o_3, \texttt{append})$	✓	✓	✓	(keine Änderung des aktuellen Sicherheitslevels)
5.	$(s_2, o_3, \texttt{write})$	×	✓	✓	ds-Eigenschaft verletzt, da $\texttt{write} \notin M_{s_2,o_3}$
6.	$(s_1, o_3, \texttt{write})$	×	✓	×	ds-Eigenschaft verletzt, da $\texttt{write} \notin M_{s_1,o_3}$, und $\star$-Eigenschaft verletzt, da $f_o(o_3) < f_c(s_1)$

Aufgabe 6. (10 Punkte) Bei dieser Multiple-Choice-Aufgabe gibt jede richtige Antwort 1 Punkt; für jede falsche Antwort wird 1 Punkt abgezogen, die Gesamtpunktzahl der Aufgabe kann jedoch nicht negativ werden. Für nicht beantwortete Fragen (kein Kreuz) werden keine Punkte abgezogen.

	wahr	falsch
Ein Public-Key-Verfahren mit deterministischem Enc-Algorithmus erfüllt den IND-CPA-Sicherheitsbegriff.		
Der One-Time-Pad ist sicher gegen Veränderungen der verschlüsselten Nachricht.		
Im CBC-Mode sind Ver- und Entschlüsselung parallelisierbar.		
$H'(x) = H(f(x))$ ist kollisionsresistent, falls H kollisionsresistent und f effizient berechenbar und injektiv ist.		
$f(k) = \frac{1}{k^c}$, für c konstant, ist vernachlässigbar in k.		
Für den erweiterten euklidischen Algorithmus $EE(a, b) = (\alpha, \beta)$ gilt $\alpha \cdot a + \beta \cdot b = 1$ für alle $a, b \in \mathbb{N}$.		
Im Kerberos-Protokoll werden keine Man-in-the-middle-Angriffe berücksichtigt.		
Für $M := \sigma^d \bmod N$ im RSA-Signaturverfahren mit $pk = (N, e), sk = (N, d)$ ist M eine Signatur für σ.		
Das Commitment $\mathsf{Com}(M; R) = \mathsf{H}(M, R)$ ist binding, falls H eine kollisionsresistente Hashfunktion ist.		
Das Chinese Wall Modell sichert eine konfliktfreie Zuordnung von Beratern zu Objekten zu.		

Lösungsvorschlag zu Aufgabe 6.

	wahr	falsch
Ein Public-Key-Verfahren mit deterministischem Enc-Algorithmus erfüllt den IND-CPA-Sicherheitsbegriff.		×
Der One-Time-Pad ist sicher gegen Veränderungen der verschlüsselten Nachricht.		×
Im CBC-Mode sind Ver- und Entschlüsselung parallelisierbar.		×
$H'(x) = H(f(x))$ ist kollisionsresistent, falls H kollisionsresistent und f effizient berechenbar und injektiv ist.	×	
$f(k) = \frac{1}{k^c}$, für c konstant, ist vernachlässigbar in k.		×
Für den erweiterten euklidischen Algorithmus $EE(a, b) = (\alpha, \beta)$ gilt $\alpha \cdot a + \beta \cdot b = 1$ für alle $a, b \in \mathbb{N}$.		×
Im Kerberos-Protokoll werden keine Man-in-the-middle-Angriffe berücksichtigt.		×
Für $M := \sigma^d \bmod N$ im RSA-Signaturverfahren mit $pk = (N, e), sk = (N, d)$ ist M eine Signatur für σ.	×	
Das Commitment $\mathsf{Com}(M; R) = \mathsf{H}(M, R)$ ist binding, falls H eine kollisionsresistente Hashfunktion ist.	×	
Das Chinese Wall Modell sichert eine konfliktfreie Zuordnung von Beratern zu Objekten zu.	×	