

H Sicherheit



1 Sicherheitsproblematik (2)

- Bedrohungen (fortges.)
 - ◆ Abstreiten von Aktivitäten (*Repudiation*)
 - Benutzer streitet Nutzung eines Dienstes ab
- Angriffsarten
 - ◆ Lauschangriff
 - unbefugtes Abhören von Kommunikation
 - ◆ Maskerade
 - Vorgeben einer falschen Identität
 - ◆ Intrigieren
 - Verfälschen von Daten
 - ◆ Wiederholung von Nachrichten
 - abgehörte Nachrichten werden erneut eingespielt (*Replay*)
 - Nachricht oft für den Angreifer unter Umständen wertlos (verschlüsselt)



1 Sicherheitsproblematik (Security)

- ★ Was ist Sicherheit (*Security*)?
- Bedrohungen (*Threats*)
 - ◆ unbefugter Zugriff auf Informationen und Dienste (*Interception*)
 - ◆ Beeinflussung der Leistungsfähigkeit eines Systems (*Interruption*)
 - Denial-of-Service-Attacke
 - Löschen wichtiger Dateien
 - ◆ unbefugte Systemveränderungen (*Modification*)
 - Veränderung von gespeicherten Daten
 - Aufzeichnen von Benutzeraktivitäten
 - ◆ unbefugtes Einbringen neuer Datensätze oder neuer Dienste (*Fabrication*)
 - Einfügen neuer Benutzerkonten
 - Aufsetzen eigener Dienste



1 Sicherheitsproblematik (3)

- Angriffsarten (fortges.)
 - ◆ Infiltration
 - eigene Komponenten werden eingeschleust
- Arten der Infiltration
 - ◆ Knacken von Passwörtern
 - Annehmen von Benutzeridentitäten
 - ◆ Virus
 - angehängt an legale Dokumente/Daten
 - wird unbeabsichtigt aktiviert
 - repliziert sich selbständig
 - enthält unter Umständen Software für weitere Bedrohungen
 - z.B. zur Zerstörung von Daten
 - z.B. zur Ausspähung von Benutzerkonten



1 Sicherheitsproblematik (4)

- Arten von Infiltration (fortges.)
 - ◆ Wurm
 - unbefugtes Einbringen und Ausführen von Programmen von außen
 - oft auf Grund von Sicherheitslöchern aber auch durch Viren
 - ◆ Trojanisches Pferd
 - angeblich sinnvoll und legitime Programme entpuppen sich als Schädling
 - warten auf unbedachte Ausführung
 - Schadenswirkung
 - können Benutzeridentitäten stehlen
 - können System für unbefugte Programme öffnen



1.1 Sicherheitsmechanismen

- Verschlüsselung (*Encryption*)
 - ◆ Daten werden verschlüsselt und damit nur für bestimmte Benutzer/ Komponenten lesbar (Vertraulichkeit, *Confidentiality*)
 - ◆ Integritätsprüfung auf unbefugte Nachrichtenänderungen (*Integrity*)
 - ◆ Grundlage für unabstreitbare Operationen (*Non-Repudiation*)
- Authentisierung (*Authentication*)
 - ◆ Überprüfung der Identität eines Benutzer, Client oder Servers
 - ◆ z.B. durch Passwort
- Autorisierung (*Authorization*)
 - ◆ Überprüfung ob authentisierter Benutzer zugreifen darf
- Auditing
 - ◆ Speichern von Benutzeraktivitäten für spätere Analyse



1 Sicherheitsproblematik (5)

- ★ Maßnahmen gegen die Bedrohungen
 - ◆ Sicherheitspolitiken
 - Festlegung, welche Operationen und Aktionen erlaubt und welche verboten sind
 - z.B. Kontobewegungen und Kontostände dürfen Unbefugten nicht bekannt werden
 - ◆ Sicherheitsmechanismen
 - Einsatz von Mechanismen zur Durchsetzung der Sicherheitspolitiken
 - z.B. Nachrichten innerhalb der verteilten Anwendung, die Kontobewegungen und Kontostände enthalten, werden verschlüsselt



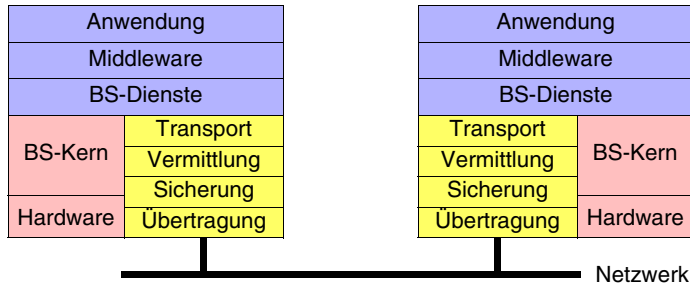
1.1 Sicherheitsmechanismen (2)

- Durchsetzung der Sicherheitspolitiken
 - ◆ Zusammenspiel der eingesetzten Sicherheitsmechanismen entscheidend
 - ◆ eine Schwachstelle kann alle anderen Maßnahmen scheitern lassen
 - z.B. ein Pufferüberlauf führt zum Root-Zugriff auf gesamten Rechner; alle anderen Maßnahmen werden wirkungslos
- ★ Konsequenz
 - ◆ umfassende Sicherheit nur schwierig erreichbar
 - ◆ koordinierte Maßnahmen erforderlich
 - ◆ diszipliniertes Arbeiten bei Umsetzung der Sicherheitsimplementierungen



1.2 Designfragen

- Hierarchischer Aufbau von Verteilten Systemen
 - ◆ logischer Aufbau in mehreren Schichten



nach Tanenbaum/van Steen

- ◆ Sicherheitsmechanismen in verschiedenen Schichten



2.1 Symmetrische Verschlüsselung (2)

- Blockorientierte Verfahren
 - ◆ Verschlüsselung einer festen Menge von Daten
 - ◆ verschiedene Verfahren für Folgeblöcke
 - ECB (Electronic Code Book): jeder Block unabhängig verschlüsselt
 - Problem: gleiche Daten ergeben gleiches Ergebnis
 - CBC (Cipher Block Chaining): nächster Block wird vor Verschlüsselung mit vorherigem verschlüsselten Block mit XOR verknüpft
 - für ersten Block ist Initialisierungsblock notwendig
- Stromorientierte Verfahren
 - ◆ byteweise Verschlüsselung von Daten
 - z.B. für Datenströme ohne feste Länge



2 Verschlüsselung

- Kryptographische Mechanismen

2.1 Symmetrische Verschlüsselung

- Funktionen
 - ◆ Verschlüsselungsfunktion E (encrypt): $E(K, T) \rightarrow C$
 - ◆ Entschlüsselungsfunktion D (decrypt): $D(K, C) \rightarrow T$
 - ◆ K = Schlüssel, T = zu verschlüsselnde Daten
- Forderungen an ein symmetrisches Verschlüsselungsverfahren
 - ◆ Wenn K unbekannt ist, soll es sehr aufwendig sein aus $E(K, T)$ das T zu ermitteln (Entschlüsselungsangriff).
 - ◆ Es soll sehr aufwendig sein, aus T und $E(K, T)$ den Schlüssel K zu ermitteln (Klartextangriff)



2.1 Symmetrische Verschlüsselung (3)

- Heute gängige Verfahren (Auswahl)

Algorithmus	Schlüssellänge (Bit)	Blocklänge (Bit)	Arbeitsweise
DES	56	64	blockorientiert
3DES	112 od. 168	64	blockorientiert
AES	128, 192 od. 256	128	blockorientiert
IDEA	128	64	blockorientiert
Blowfish	variabel bis 448	64	blockorientiert
SEAL	160	8	stromorientiert
RC4	variabel bis 2048	8	stromorientiert



2.2 Einsatz symmetrischer Verfahren

- Vertraulichkeit von Daten
 - ◆ Verschlüsselung von Daten für Ablage an nicht vertrauenswürdigen Orten
 - z.B. Festplatte in Multi-User-System
 - Schlüssel nur einem Benutzer/einer Komponente bekannt (Passwort)
 - ◆ Verschlüsselung von Daten bei Kommunikation
 - beide Kommunikationspartner müssen Schlüssel kennen
 - andere dürfen Schlüssel nicht kennen
- Authentisierung
 - ◆ Challenge-Response-Protokoll
 - beide Kommunikationspartner müssen Schlüssel kennen
 - andere dürfen Schlüssel nicht kennen



2.3 Challenge-Response-Protokoll

- Aufbau eines sicheren Kanals
 - ◆ zwischen authentisierten Komponenten
 - ◆ keine Verfälschung von Nachrichten
 - ◆ keine zusätzlichen Nachrichten
 - ◆ keine Nachrichtenverluste
- Beispielsweise: Aufbau einer TCP/IP-Verbindung
 - ◆ Annahme bei stehender Verbindung
 - keine Verfälschung möglich
 - würde Abfangen und Neueinspielen erfordern
 - keine zusätzlichen Nachrichten möglich
 - verhindert durch Sequenznummern
 - keine Nachrichtenverluste möglich
 - verhindert durch Sequenznummern



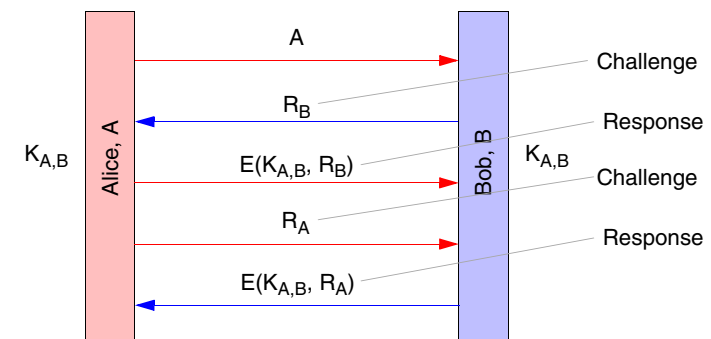
2.2 Einsatz symmetrischer Verfahren (2)

- Integrität der Nachricht
 - ◆ Sicherstellung dass Nachricht unverfälscht kommuniziert wurde
 - ◆ Verschlüsselung der Nachricht reicht aus
 - verfälschte Nachricht lässt sich nicht entschlüsseln
 - erkennbare Nachrichtenstruktur erforderlich
 - z.B. Sequenznummer
 - ◆ Verschlüsselung eines Digests (digitale Signatur)
 - sichere Hash-Funktion erforderlich



2.3 Challenge-Response-Protokoll (2)

- Authentisierung der beiden Komponenten



- ◆ B weiß, dass Kommunikationspartner A ist (nur er hat noch $K_{A,B}$)
- ◆ A weiß, dass Kommunikationspartner B ist (nur er hat noch $K_{A,B}$)



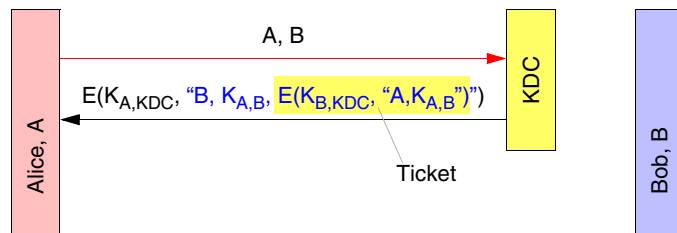
2.3 Challenge-Response-Protokoll (3)

- Anschließender Datenaustausch über Verbindung
 - ◆ Verschlüsselung der Daten mit $K_{A,B}$
 - ◆ Ausschluss von Verfälschung, Einfügung und Verlust von Nachrichten
- Anfälligkeit des Challenge-Response-Protokolls
 - ◆ Mithören: Angreifer erfährt Challenge-Response-Paare
 - Klartextangriff: Versuch Schlüssel $K_{A,B}$ zu ermitteln
 - falls Werteraum zu klein, Wahrscheinlichkeit für wiederholte Verwendung gleicher Challenges groß
 - ◆ Einfügen, Modifizieren und Herausnehmen von Nachrichten führt nicht zum Ziel
 - außer Denial-of-Service-Angriff



2.4 Schlüsselverwaltung (2)

■ Needham-Schröder-Protokoll



- ◆ Anfrage and KDC über Verbindung von A nach B
- ◆ Ergebnis: nur für A entzifferbare Nachricht
 - enthält generierten Schlüssel $K_{A,B}$ und Ticket
- ◆ Ticket ist nur für B lesbar und enthält ebenfalls $K_{A,B}$



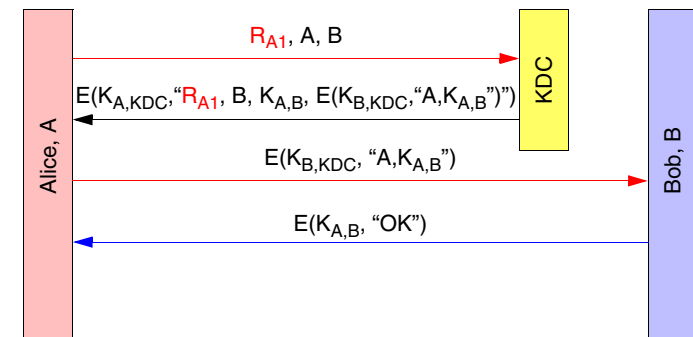
2.4 Schlüsselverwaltung

- Problem
 - ◆ bei N Kommunikationspartner sind $(N^2-N)/2$ Schlüssel notwendig
- Lösung Schlüsselverteilungsdienst (*Key Distribution Center, KDC*)
 - ◆ alle Teilnehmer haben einen geheimen Schlüssel, den das KDC kennt
 - ◆ bei N Kommunikationspartner sind N Schlüssel notwendig
- Vorgehensweise
 - ◆ KDC generiert einen Schlüssel für einen Verbindungswunsch
 - ◆ Schlüssel wird auf sicherem Weg an beide Partner übertragen



2.4 Schlüsselverwaltung (3)

■ Needham-Schröder-Protokoll

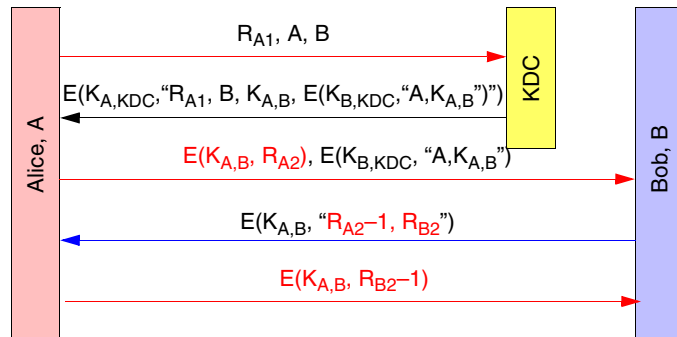


- ◆ Problem 1 (A-KDC): Zuordnung der Nachrichten nicht gesichert
 - Wiedereinspielung alter Nachrichten möglich
 - Hinzufügen einer zufälligen Identifikation (*Nonce*)



2.4 Schlüsselverwaltung (4)

■ Needham-Schröder-Protokoll



- ◆ Problem 2 (A-B): Zuordnung der Nachrichten nicht gesichert
 - Wiedereinspielung alter Nachrichten möglich
 - Hinzufügen von Challenge-Response-Protokollteilen



2.4 Schlüsselverwaltung (5)

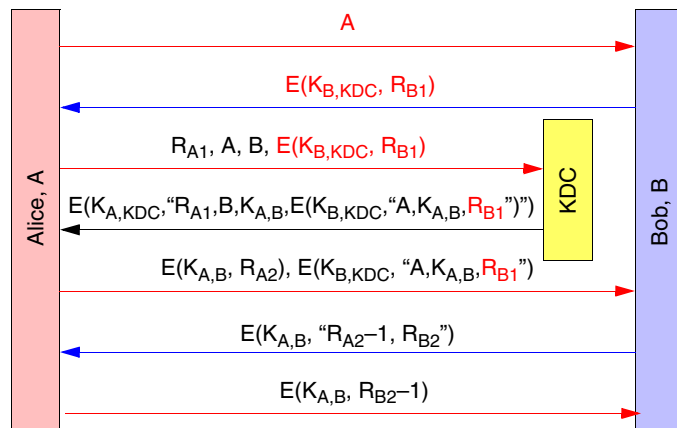
■ Needham-Schröder-Protokoll

- ◆ Versenden von R_{A2} oder R_{B2} allein reicht nicht
 - könnte abgefangen und wieder eingespielt sein
- ◆ daher: Veränderung der Challenge (-1) wird Response
- ◆ Weiteres Problem: Entdeckung eines alte $K_{A,B}$
 - Wiedereinspielung des Verbindungswunsches (3. Nachricht) möglich
 - B würde mit Angreifer Sitzung aufbauen
 - Verknüpfung der KDC-Anfrage mit Verbindungsaufbau notwendig



2.4 Schlüsselverwaltung (6)

■ Korrektes Needham-Schröder-Protokoll



2.4 Schlüsselverwaltung (7)

■ Ausgefeiltere Schlüsselverwaltung

- ◆ Realisierung von $K_{A,KDC}$ über Benutzerpasswort
 - bei jeder Dienstaufnahme muss Passwort eingegeben werden
 - Lösung: Ticket-Granting-Server stellt temporäres Ticket aus
 - gilt ohne Passwort als Schlüssel für Verbindungsaufnahme mit KDC
- ◆ große Systeme benötigen mehrere KDCs
 - Lösung: Ticket-Granting-Server erstellt Tickets für bestimmten KDC
- ◆ Implementierung: [Kerberos](#)
 - auch in Windows-Active-Directory-Authentication enthalten



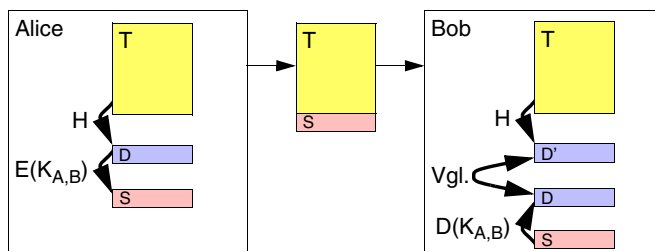
2.5 Sichere Hash-Funktion

- Funktion
 - ◆ Hashfunktion: $H(\text{hash}): H(T) \rightarrow D$
 - ◆ T = zu behandelnde Daten beliebiger Länge
 - ◆ D = Hash-Wert fester Länge (Digest)
- Forderungen an sichere Hash-Funktionen
 - ◆ Die Funktion H ist einfach zu berechnen.
 - ◆ Für gegebenen Wert D ist es nahezu unmöglich ein T zu finden, so dass gilt: $H(T) = D$ (Einwegigkeit)
 - ◆ Für ein T ist es nahezu unmöglich ein S zu finden, so dass gilt $H(T) = H(S)$ (Schwache Kollisionsresistenz)
 - ◆ Es ist nahezu unmöglich ein Paar (T, S) zu finden, so dass gilt $H(T) = H(S)$ (Starke Kollisionsresistenz)



2.6 Digitale Signatur

- Einsatz von sicheren Hash-Funktionen für digitale Signaturen
 - ◆ Nachricht wird im Klartext versendet
 - ◆ Hashwert wird verschlüsselt angehängt (Signatur)
 - über Verschlüsselung wird bewiesen, wie Nachricht beim Verschlüssler ausgesehen hat (Integrität)



- Vergleich testet Signatur und damit Integrität der Nachricht



2.5 Sichere Hash-Funktionen (2)

- Heute gängige Verfahren (Auswahl)

Algorithmus	Hashlänge (Bit)	Bemerkung
SHA-1	160	Länge von $T < 2^{64}$ Bits
MD5	128	beliebige Länge
RIPEMD-160	160	beliebige Länge



2.7 Digitale Signatur (2)

- Digitale Signatur als Authentisierung
 - ◆ Kann nur einer den Schlüssel für korrekte Signatur besitzen ist gleichzeitig Nachricht authentisiert.
- Authentisierung ohne Verschlüsselung
 - ◆ gemeinsames Geheimnis zwischen zwei Partnern, G

