IT-Sicherheit

Schlüsselverwaltung

Prof. Dr. Dominik Merli, Prof. Dr. Lothar Braun

Sommersemester 2020

Hochschule Augsburg - Fakultät für Informatik

Schlüssel (engl. key)

Wozu Schlüssel?

- Laut Kerckhoffs' Prinzip
  - Schlüssel ist einziges Geheimnis in Krypto-Algorithmus
- Unbedingt notwendig für …
  - Ver-/Entschlüsselung
  - MAC-Generierung/-Verifikation
  - Signatur-Generierung/-Verifikation
  - Authentifikation

Schlüssel mit langer Gültigkeitsdauer

- Übliche Bezeichnungen
  - Master Key
  - Root Key
  - Long-Term Key
- Eigenschaften
  - Schlüssel wird einmal generiert und dann gespeichert
  - Kein weiterer Aktualisierungs-Aufwand nötig
  - ABER: Schlüssel hat hohen Schutzbedarf!
- Oberstes Ziel: Vertraulichkeit

Schlüssel mit kurzer Gültigkeitsdauer

- Übliche Bezeichnungen
  - Session Keys
  - Ephemeral Keys
  - Short-Term Keys

- Eigenschaften
  - Weniger Schaden, wenn Schlüssel bekannt wird
  - Weniger Daten abhängig von einem Schlüssel
  - Angreifer benötigt evtl. mehrere Schlüssel für Erfolg
  - Schutzbedarf geringer
  - ABER: Regelmäßiger Aktualisierung notwendig!

- Ziel: Key Freshness

Implementierung von Key Freshness

- Regelmäßiger Schlüsselaustausch
  - Möglich, aber evtl. zu aufwändig
  - Gilt speziell für asymmetrische Kryptographie

- Schlüsselableitung
  - Ableitung mit Key Derivation Function (KDF)
  - Input sind Master Key und öffentlicher Parameter
  - Abgeleitete Schlüssel verraten nichts über Master Key

Schlüsselableitung auf Basis von Schlüsseln

- Key Derivation Function (KDF)
  - Verarbeitet Schlüssel und öffentlichen Parameter
  - z.B. auf Basis von HMAC ($\rightarrow$ HKDF) oder auch Block-Ciffren
- Master Key
  - z.B. Schlüssel in HMAC-SHA256
  - z.B. Schlüssel für AES Verschlüsselung
- Öffentlicher Parameter
  - z.B. Nonce, generiert von einem Kommunikationspartner
  - z.B. Zählerwert, der ständig aktualisiert wird (Datum)

Schlüsselableitung auf Basis von Passwörtern

- Password-Based Key Derivation Function Version 2 (PBKDF2)
  - Basiert auf HMAC
  - Verarbeitet Passwort und Salt, um Schlüssel abzuleiten
  - Gewünschte Schlüssellänge kann eingestellt werden
- Zusätzliche Anforderung
  - Aufwand zum Brute-Forcing soll möglichst hoch sein
  - $\rightarrow$ Möglichst hohe Anzahl an Iterationen einstellbar
- Passwort-Speicherung mit `scrypt`
  - Entworfen, um sichere Passwort-Speicherung zu ermöglichen
  - CPU und Memory-Kosten für Angreifer einstellbar

Für viele Anwendungen nötig: Schlüsselaustausch

- Zueinander passende Schlüssel für Kommunikationspartner
- Austausch über …
  - sicheren Kanal, z.B. persönliches Treffen
  - unsicheren Kanal, z.B. Internet
- Verfahren
  - Key Transport: Ein Partner generiert und verteilt Schlüssel
  - Key Agreement: Partner generieren Schlüssel zusammen

Schlüsselverwaltung mit symmetrischer Kryptographie

Schlüsselverteilungsproblem

- Netzwerk mit n Teilnehmern
  - Jeder will mit jedem kommunizieren
  - Jedes Kommunikations-Paar benötigt eigenen Schlüssel

- Anzahl an Schlüssel
  - Jeder Teilnehmer muss $n - 1$ Schlüssel speichern
  - Insgesamt $n(n-1) \approx n^2$ Schlüssel im Netzwerk
  - $n(n-1)/2$ einzigartige Schlüsselpaare

- Probleme
  - Neue Teilnehmer benötigen sichere Verbindungen zu allen anderen Teilnehmern
  - Gesamtzahl der Schlüssel wächst stark mit Teilnehmern
    (z.B. $n = 10 \rightarrow 90$ Schlüssel, $n = 1000 \rightarrow 999000$ Schlüssel)

Idee

- Sicherer Kanal nur notwendig bei neuem Teilnehmer
  - z.B. (manuelle) Installation durch Administrator
  - z.B. Schlüsseleinbringung während der Herstellung
- Symmetrische Kryptographie eignet sich für Key Transport

Key Distribution Center (KDC)

- Zentraler Server
  - Vollkommen vertrauenswürdig
  - Teilt sich mit jedem Teilnehmer einen Schlüssel für Key Transport → Key Encryption Key (KEK)
  - KEK wurde über sicheren Kanal geteilt
- Kann Anfragen nach Session Keys für Teilnehmer beantworten

Grundlegendes KDC Protokoll

- Beispiel: Alice will mit Bob kommunizieren
  - Alice teilt sich k_A mit KDC
  - Bob teilt sich k_B mit KDC

- Anfrage für Kommunikation (Alice - Bob) an KDC
  - KDC generiert zufälligen Schlüssel k_{ses}
  - Nachricht an Alice: $y_A = encr_{k_A}(k_{ses})$
  - Nachricht an Bob: $y_B = encr_{k_B}(k_{ses})$

- Nach Entschlüsselung: Alice und Bob teilen sich k_{ses}
  - Alice: $k_{ses} = decr_{k_A}(y_A)$
  - Bob: $k_{ses} = decr_{k_B}(y_B)$
  - k_{ses} verwendbar für symmetrische Kryptographie

Eigenschaften des KDC Protokolls

- Zwei verschiedene Schlüsselarten
  - Langzeit-Schlüssel k_A und k_B (engl. long-term keys)
  - Kurzlebiger Session Key k_{ses}

- Sicherheit
  - Schützt gegen Abhören durch passiven Angreifer
  - Aber Replay Angriffe möglich:
    - Angreifer bekommt Zugriff auf alten k_{ses} und zugehörige y_A und y_B
    - Angreifer imitiert KDC und schickt y_A und y_B an Alice und Bob
    - Angreifer kann Kommunikation belauschen
  - Auch Man-in-the-Middle Angriff auf Anfrage möglich

Kerberos

- In den 80er Jahren am MIT entwickelt
- Version 5 definiert in RFC 4120
- Einsatz bei MS Windows zur Authentifizierung im Netzwerk

- Timeliness Eigenschaft
  - KDC definiert eine Lebenszeit T für einen Session Key
  - Anfrage von Alice enthält Zeitstempel $\rightarrow$ Bob sieht, ob aktuell

- Authentifikation und Key Confirmation
  - Alice schickt Nonce zu KDC und überprüft Nonce in Antwort
    (Alice kann KDC Antwort verifizieren)
  - Identität der Teilnehmer in KDC Antwort enthalten
    (Alice und Bob können sich gegenseitig verifizieren)

Nachteile symmetrischer Schlüsselverteilung

- Kommunikation
  - Kontakt zu KDC nötig für neuen Session Key
- Initialisierung
  - Sicherer Kanal für KEK Austausch notwendig
- KEK Datenbank
  - Alle geheimen Schlüssel liegen an einem zentralen Punkt
- Keine Perfect Forward Secrecy (PFS)
  - Wenn KEK bekannt, bisherige und zukünftige Kommunikation unsicher

QUIZ: Schlüsselaustausch (twbk.de)

Die Gültigkeitsdauer eines Schlüssels ist entscheidend für dessen Risiko- und Schutzbedarfsbewertung. Welche Vorteile bieten kurzlebige Schlüssel?

A) Weniger Datenverarbeitung pro Schlüssel und somit weniger Schaden bei Kompromittierung

B) Geringere Kosten, da weniger Rechenaufwand mit kurzlebigen Schlüsseln

C) Keine Vorteile! Sollten vermieden werden, wenn möglich!

Stellen Sie sich vor, Sie möchten ein Key Distribution Center angreifen, um möglichst viele Konversationen im Netzwerk über eine möglichst lange Zeit zu überwachen. Nach kurzer Beobachtung stellen Sie fest, dass Sie entweder eine Datenbank mit Schlüsseln k_A, k_B, etc. entwenden können oder aber die aktuelle Datenbank mit allen k_{ses} Schlüsseln. Beides wäre zeitlich nicht möglich. Für was entscheiden Sie sich?

A) k_A, k_B, etc.

B) Alle k_{ses} ...

Schlüsselverwaltung mit asymmetrischer Kryptographie

Ideen

- Öffentliche Schlüssel sind nicht vertraulich
  - Sollte Schlüsselmanagement deutlich vereinfachen
- Key Transport
  - Session Key wird verschlüsselt und an Teilnehmer verteilt
  - z.B. mit RSA Verschlüsselung
- Key Agreement
  - Session Key wird zusammen per Protokoll generiert
  - z.B. mit Diffie-Hellman

Diffie-Hellman Schlüsselaustausch

- Veröffentlicht 1976 von Whitfield Diffie und Martin Hellman
- Basis vieler Sicherheitsprotokolle
  - z.B. Secure Shell (SSH), Transport Layer Security (TLS) und IPSec
- Basiert auf Discrete Logarithm Problem in zyklischer Gruppe $\mathbb{Z}_p^*$

Diffie-Hellman Protokoll

- Öffentliche Domain Parameter
  - Große Primzahl p
  - Primitives Element $\alpha \in \{2, 3, ..., p-2\}$
- Vorbereitung (Schlüsselgenerierung)
  - Alice wählt $a = k_{priv,A} \in \{2, 3, ..., p-2\}$
  - Alice berechnet $A = k_{pub,A} = \alpha^a \ mod \ p$
  - Bob wählt $b = k_{priv,B} \in \{2, 3, ..., p-2\}$
  - Bob berechnet $B = k_{pub,B} = \alpha^b \ mod \ p$
  - Alice und Bob tauschen $A = k_{pub,A}$ und $B = k_{pub,B}$ aus
- Schlüsselaustausch
  - Alice berechnet $k_{AB} = B^a \ mod \ p = \alpha^{ba} \ mod \ p$
  - Bob berechnet $k_{AB} = A^b \ mod \ p = \alpha^{ab} \ mod \ p$
  - k_{AB} ist gemeinsames Geheimnis

Kritisches Problem

- Man-in-the-Middle Angriff
  - Angreifer sitzt zwischen Alice und Bob
  - Angreifer ersetzt Public Keys durch eigenen Public Key
  - Alice und Bob vereinbaren Schlüssel mit Angreifer
  - Angreifer kann Nachrichten lesen ohne bemerkt zu werden

- Wichtige Forderung: authentifizierter Kanal
  - Gegenseitige Public Key Verifikation von Alice und Bob
  - Manipulationen der Schlüssel würden bemerkt werden

- Lösung: **Zertifikate** zur Authentifizierung der Schlüssel

Zertifikate

- Mindestens
  - Identität und Public Key eines Teilnehmers
  - Signatur der Daten durch eine vertrauenswürdige Stelle
- Vorteil
  - Integrität und Authentizität von Public Keys verifizierbar
  - Signatur-Stelle für Zertifikate muss vertrauenswürdig sein
- Wichtiger Standard für Zertifikate: X.509

X.509 Struktur

- Certificate
  - Version Number
  - Serial Number
  - Signature Algorithm ID
  - Issuer Name
  - Validity period
    - Not Before
    - Not After
  - Subject name
  - Subject Public Key Info
    - Public Key Algorithm
    - Subject Public Key
  - Issuer Unique Identifier (optional)
  - Subject Unique Identifier (optional)
  - Extensions (optional)
- Certificate Signature Algorithm
- Certificate Signature

Certificate Authority (CA)

- Zertifikats-Generierung
  - Benutzer schickt Certificate Signing Request (CSR) an CA
  - Enthält mindestens Identität und Public Key des Nutzers
  - CA verifiziert Identität, signiert Daten und stellt Zertifikat aus
  - **Wichtig:** CA hat keinen Zugriff auf Private Key des Nutzers

- Weltweit jede Menge an CAs
  - Regierungen
  - Kommerzielle Anbieter
  - Unternehmen (intern)

- Internet: CA/Browser Forum $\rightarrow$ Public Keys in Browsern

Kombination von CAs

- Verkettung von Zertifikaten
  - CA1 signiert Zertifikat von CA2
  - CA2 signiert Zertifikat von Alice
  - Verifikation: rückwärts durch Zertifikatskette
  - Oberstes Zertifikat: Wurzel-Zertifikat (engl. root certificate)

- Cross-Signing
  - CAs können sich gegenseitig zertifizieren
  - Erleichtert Verifikation von Zertifikaten anderer CAs

Revozierung von Zertifikaten

- Mögliche Probleme mit Zertifikaten
  - Smartcard mit Private Key wird verloren
  - Angestellter verlässt Unternehmen
  - Schlüssel wird kompromitiert

- Certificate Revocation List (CRL)
  - Enthält widerrufene Zertifikate
  - Muss von CA signiert werden
  - Problem: Jede Verifikation erfordert Verbindung zur CA
  - Aber: Periodische Versendung führt zu Zeitversatz

Public Key Infrastruktur (PKI)

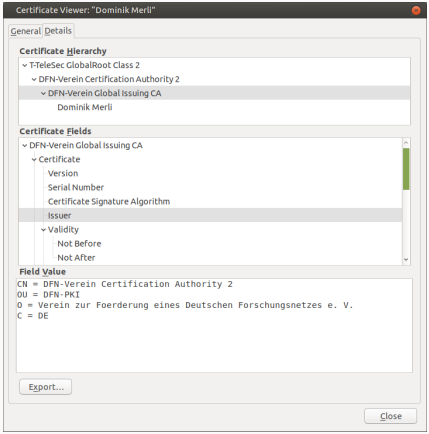
- Prozesse und Organisation
  - Certificate Authority (CA): signiert Zertifikate
  - Registration Authority (RA): verifiziert Zertifikatsanfragen
  - Directory Service: indiziert Liste von Zertifikaten
  - Viele weitere Dienste möglich ...
- Sehr aufwendiger Prozess: Zertifizierung, etc.

Kostenlose CA: Let's Encrypt

- Gegründet im April 2016
- Finanziert durch Spenden und Sponsoren
- Stellt kostenlose Zertifikate für Websites aus
- Ziel: Möglichst weite Verbreitung von HTTPS
- Bisher großer Erfolg
- Link: https://letsencrypt.org/

CA der Hochschule Augsburg

- Zertifizierungsstelle an der HSA
  - Link:
    https://www.hs-augsburg.de/
    Rechenzentrum/ZertifizierungCA.html
- Stellt Zertifikate für
  E-Mail-Nutzer und Server aus
- Basiert auf der PKI des
  Deutschen Forschungsnetztes (DFN)



Aufgabe

Analysieren Sie die Zertifikate von Webseiten!

z.B.
www.google.com
www.heise.de
www.mail.ru
...

Gibt es noch Fragen?