Hardware Security Modules

Senf

mail@senf.space, GPG: 0xa3ba9353

Datenspuren Dresden
2025-09-20 14:15-15:30 UTC+2
kleiner Saal

1. Kryptographische Betrachtungen
2. Protokolle
3. HSM allgemein
4. Implementierungen von HSM
5. Nutzung von HSM

Begleitung zum Vortrag: Workshop/Cryptoparty

2025-09-21 15:15-17:00 UTC+2, Kabinett

- Gemeinsame Einrichtung von Nitrokey 3A
  - Ich richte mich nach Wünschen, habe noch nichts vorbereitet
- Gegenseitiges Signieren von (frisch erstellten) PGP-Schlüsseln
  - Abgleich: Schlüsselpaar (durch FP) mit Person (meist durch Ausweis)
  - Unterschreibende Person entscheidet selbst, insbesondere bei Pseudonymen
  - Name und Fingerprints können mir im Voraus geschickt werden: mailto:senf@c3d2.de

Weiterer Vortrag

Nitrokey, Simon, 2025-09-20 16:00-16:45 UTC+2, Kabinett

- FIDO2 Level 1/2 Zertifizierung
- FIDO2 Post-Quantum Cryptography
- Stand von FIDO2 in Webbrowsern und Android
- Vergleich von Algorithmen (z.B. ECC- vs. RSA-Implementierung)
- Coreboot

Vorwissen

Die folgenden Folien beinhalten etwas Vorwissen, um sich im Vortrag eine allenfalls *halbgare* Vorstellung von HSM zu erarbeiten, und nicht komplett ratlos dabei zu sein.

Fragen können zum Beginn des Vortrags gestellt werden.

Empfehlungen im Vorfeld

Wer Lust und Zeit hat, sollte sich folgende Vorträge ansehen:

- GPN23: Kryptographie-Hardware: Von TPMs bis Nitrokeys
- 35c3: Wallet Security, How (not) to protect private keys
- CoSin 2019: OpenPGP-Smartcard-Vergleich
- CLT25: SSH-Anmeldung mit GnuPG und SmartCards
- CLT25: Passwortlose Logins mit PassKeys
- 32c3: Hardware-Trojaner in Security-Chips
- Openchaos 2015: Kryptographisch sicherer Zufall
- 31C3: ECCHacks

Schutzziele: Was soll erreicht werden?

- Vertraulichkeit
  Lesen nur durch Autorisierte gestatten
- Integrität
  Alle Änderungen müssen nachvollziehbar sein
- Zurechenbarkeit
  Handlung kann genau einer Person zugeordnet werden
- Verbindlichkeit
  Handlung kann nicht abgestritten werden
- Verfügbarkeit
  Zugriff auf Daten muss gewährleistet sein

Schutzziele: Was soll noch erreicht werden?

Im Gegensatz hierzu stehen aber auch:

- Anonymität
  Handlung kann nicht genau einer Person zugeordnet werden
- Pseudonymisierung
  Substitution von Erkennungsmerkmalen
- Verdecktheit
  Existenz von Inhalten darf nicht erkennbar sein

Vorüberlegung: Was lässt sich mit HSM davon erreichen?

Begriffsklärung: Sicherheit

Nachrichten: M; Schlüssel: K, Geheimtexte: C

- Informationstheoretische Sicherheit
  - Amgreifer unbeschränkt (in Rechenleistung und Speicher)
  - Kein besserer Angriff als stumpfes Raten
  - Wissen über Nachrichten vor und nach Beobachtung gleich: $\forall m \in M, \forall c \in C : p(m|c) = p(m)$
  - Notwendigkeit: $|K| \geq |C| \geq |M|$
  - Beispiel: One-Time-Pad

Begriffsklärung: Sicherheit

- Semantische Sicherheit
  - Asymptotische Betrachtung; Angreifer polynomiell beschränkt
  - Ununterscheidbarkeit: Keine Relation $m_1 \rightarrow c_1, m_2 \rightarrow c_2$ herstellbar
- Non-Malleability
  - Angreifer soll Klartext nicht umformen können

Kerckhoffsches Prinzip

aus *La Cryptographie militaire*, 1883

- Keine effiziente Entschlüsselung ohne Schlüssel
- Auch nicht, wenn Algorithmus bekannt wird
- Schlüssel muss leicht tauschbar sein
- System muss transportabel sein
- System sollte gut untersucht worden sein
  $\rightarrow$ setzt Untersuchbarkeit voraus

Mehrfaktorauthentifizierung

- Faktoren: Wissen, Besitz, Biometrie/Inhärenz
  - Wissen, Besitz: austauschbar, Kompromittierbarkeit schwankt
  - Biometrie: nicht austauschbar, leicht kompromittierbar
- Nutzung mehrerer Faktoren unterschiedlicher Ausprägung
- Beispiel: Smartcard; private key mit PIN verschlüsselt

Symmetrische Kryptographie

- Tupel (M, C, K, E, D); $E : K \times M \to C$; $D : K \times C \to M$
- Methoden: Transposition, Substitution
- Ziele: Konfusion, Diffusion (nach Claude Shannon)
- Historisch: Caeser, Vignere, Enigma, etc.
- Erstes modernes, verbreitetes System: DES (1976, IBM/NSA)
  - Blockchiffre: Aufteilung von M in $64b$-Blöcke
  - Durchlaufen von 16 Runden mit generierten Teilschlüsseln
- Heute: Rijndael (AES), Twofish, CAST, Serpent

Betriebsmodi

- Symmetrische Systeme oft Blockchiffren
- Electronic Code Book (ECB):
  $m_1 = m_2 \leftrightarrow enc(m_1, k) = enc(m_2, k)$
- Besser: Verkettung oder Einfluss von Position
  - Cipher Block Chaining (CBC): $c_i = enc(m_i \oplus enc(m_{i-1}, k), k)$
  - Counter Mode (CTR): $c_i = m_i \oplus enc(T_i, k)$, $T_i = T_{i-1} + 1$
  - Zufälliger Initialisierungsvektor T_0 (auch `IV`) nötig

Betriebsmodi

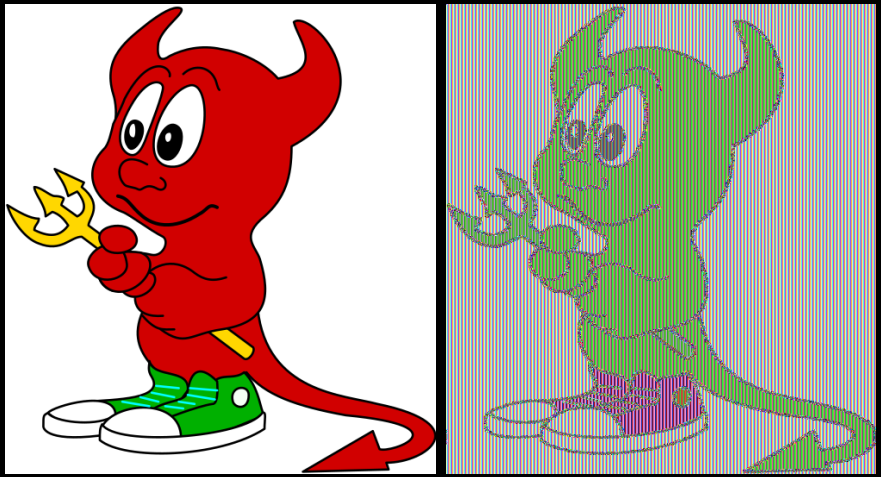

Abbildung 1: links: Quelldatei; rechts: nach `openssl enc -aes-128-ecb`

Asymmetrische Kryptographie

- Tupel (M, C, K_E, K_D, E, D);,
  $E : K_E \times M \to C$ (c muss nicht eindeutig sein!),
  $D : K_D \times C \to M$
- Grundlage: Einweg-Trapdoor-Funktionen
  - Berechnung des Funktionswertes einfach (polynomielle Laufzeit)
  - Berechnung des Urbildes schwierig (kein Angriff in P vorhanden)
  - Urbild unter Kenntnis weiterer Parameter einfach berechenbar
- Ziel: Schlüsselaustausch vereinfachen
- Auch nett: oftmals zur Signatur nutzbar
- Erste Forschung: 1974, Merkles Puzzle

Hybride Verschlüsselung

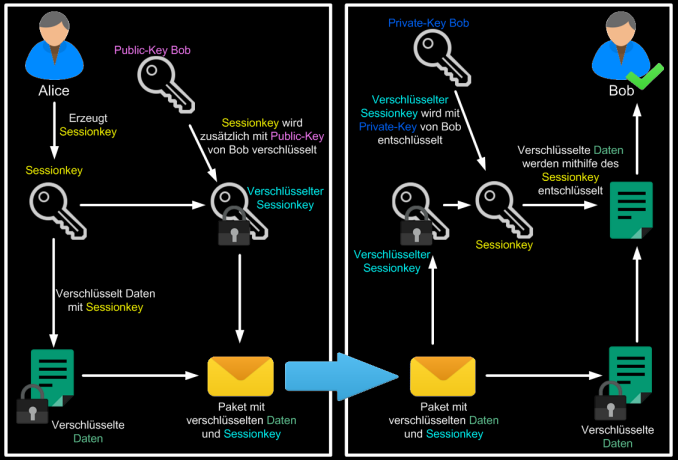

Abbildung 2: https://commons.wikimedia.org/wiki/File:Hybride_VerschlÃsselung.png

RSA-Kryptosystem

- Rivest, Shamir, Aldeman; MIT; 1977)
- Generierung zweier geheimer *Strong primes* p, q, wobei $\frac{(p-1)}{2}$ und $\frac{(q-1)}{2}$ auch prim
- Öffentlich: $n = p \cdot q$, Faktorisierung schwierig
- Pubkey k_e zufällig aus $1 < k_e < \Phi(n),\ ggT(k_e, \Phi(n)) = 1$
- Privkey: $k_d = k_e^{-1}\ mod\ \Phi(n)$
- $enc : c = (r, m, hash(r, m))^{k_e}\ mod\ n$, Hinzufügen von Zufall und Redundanz reduziert passive Angriffsmöglichkeiten
- Auch zur Signatur nutzbar

Diffie-Hellman(-Merkle)-Schlüsselaustausch

- Serverseitig vorgegeben: p (prim), $g \in \mathbb{Z}_p^*$
- Alice: $A = g^a \ mod \ p$; Bob: $B = g^b \ mod \ p$, a, b bleiben geheim
- Austausch von A, B über öffentlichen Kanal
- Alice: $K_a = B^a \ mod \ p$; $K_b = A^b \ mod \ p$
- gemeinsamer Schlüssel: $K_a = K_b$
- Man-in-the-Middle-Angriff: Mallory als *Proxy*
  - Abwehr/Erkennung durch Signatur von g, A, B
  - Abwehr/Erkennung durchi Nutzung eines zweiten Kanals nach KE
- Ähnlich läuft ElGamal ($\rightarrow$ DSA) ab

Elliptic Curve Cryptography

- Miller, Koblitz (1985)
- EC: Menge aller Punkte in $y^2 = x^3 + ax + b,\ x, y \in \mathbb{R}$
- DL lässt sich statt in $\mathbb{Z}^*$ auch auf nicht-singuläre EC ausführen
  - Voraussetzung: $4a^3 + 27b^2 \neq 0$
  - Mit Operation $+$ abelsche Gruppe
- Angriff mit Indexkalkül-Methode nicht durchführbar
  $\rightarrow$ Nutzung deutlich kleinerer Schlüssel (z.B. $256b$) möglich
- Siehe auch Wallet Security #t=240

Standardisierung einiger Kurven

Siehe auch: https://safecurves.cr.yp.to/rigid.html

- NIST-Kurven (auch secp)
  - Von NSA entwickelt
  - *Coefficients generated by hashing unexplained seeds*
- Brainpool-Kurven: von BKA/BSI (https://bada55.cr.yp.to/brainpool.html)
- *Akademische* EC: djb, Lange: SafeCurves (ed25519 ($mod\ 2^{255} - 19$), ed448)

Post Quantum Cryptography

- Peter Shors Algorithmus (1994)
- Faktorisierung und DLP: Reduktion auf polynomiell
- Benötigt Quantencomputer (ca. $10^7 \ldots 10^8$ Qubits)
- Aber: Geheimdienste horten vermutlich Geheimtexte
- Anwendung: `sntrup761x25519-sha512@openssh.com`, GnuPG 2.5.0 T6815: Kyber+ECC

Zufallszahlen

- Notwendigkeiten:
  - Erzeugung von lange genutzten Schlüsseln (Keypairs, PSK)
  - Erzeugung von Sitzungsschlüsseln
  - Erzeugung von Nonces (für RSA, ElGamal)
- Optimal: Nicht-deterministisch, Nutzung physikalischer Prozesse:
  - Rauschen von Bauteilen (z.B. CMOS-Sensor), Metastabilität von Flipflops
  - Atmosphärenrauschen, radioaktive Zerfallszeiten
- NTG-1: non-physical true RNG
  `/dev/random@linux`: LFSR aus I/O gefüttert und mit SHA-1 gemischt
- PRNG (Pseudo-nd): echter Zufall als Startwert für Reihen
- Weiterführend: Openchaos 2015: Kryptographisch sicherer Zufall

Kryptographische Hashfunktionen

- Nicht-injektive, deterministische Einwegfunktion
- Hashwert hat feste Länge, unabhängig vom Eingabewert
- Kollisionsresistenz: $m_1, m_2 : h(m_1) = h(m_2)$ nur schwierig zu finden
- Diverse Algorithmen, oft Verkettung von Blockchiffren
- Nutzung: Integritätsprüfung von Nachrichten, PRNG, key derivation (Speicherung oder Verwendung im Kryptosystem)
- Verwendung von *Salt* verhindert Angriff mit vorausberechneten *Rainbow-Tables*

Pretty Good Privacy, OpenPGP

- Hybrides Kryptosystem, nutzt ElGamal, RSA oder ed25519
- Nutzung: Absichern von E-Mail (PGP/MIME), XMPP; Signieren von Dokumenten, Code ($\rightarrow$ `git`), Binaries ($\rightarrow$ `apt`), ...
- Schlüsselgenerierung autonom
  - Verifikation anhand von FP-Vergleich über weiteren Kanal
  - Web of trust: Signieren von Pubkeys; Vertrauensketten bilden
- Erste Version: Phil Zimmermann, 1991; Quelltext in Buch exportiert; heute Symantec-Gedöns(?)
- Standardisierung: *OpenPGP*, diverse RFC, viele alte Algorithmen
- Meistgenutzte freie, *aktuelle* Implementierung: GnuPG
  (NetPGP scheint irgendwie eingeschlafen zu sein)

Transport Layer Security

- Hybrides Kryptosystem, diverse Algorithmen/Implementierungen
- Frühere Bezeichnung: Secure Sockets Layer (SSL)
- Standardisierung zur sicheren Anwendung von Kryptosystemen
- Nutzung von ausgestellen X.509-Zertifikaten
  - Nutzer generiert Privkey und dazugehörige Request
  - Hierarchie: Authority stellt Cert zu eingereichter CSR aus
  - Andere Nutzer vertrauen Cert, wenn Signatur von vertrauenswürdiger CA korrekt ist (und noch gültig ist und Adresse und Zweck passen)
- Anwendung u.a. in HTTP**S**, OpenVPN, S/MIME