Willkommen bei Verteilte Systeme!

Von Datenbanken
über Webdienste
bis zu p2p und Sensornetzen.

:)

Heute: **Sicherheit (danach Rückblick)**.

Ablauf heute

- Sicherheit

```
--- PAUSE 14:15 ---
```

- Rückblick und Klausurvorbereitung

Vorweg: Absolut Sichere Systeme

1. Völlig vertrauliche Schnittstelle
2. Alles sofort vergessen

aber

1. nie völlig vertraulich
2. nutzlos — und schwierig

⇒ keine absolute Sicherheit, aber Näherungen

Beispiel: Letterblock Diceware

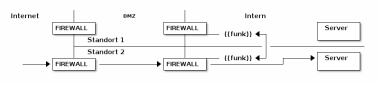
Letterblock Diceware

www.draketo.de/software/letterblock-diceware

	1	2	3	4	5	6
1	1	A	J	a	h	px
2	26	BC	LR	bc	i	r
3	37	DH	N	d	j	t
4	48	E	PX	e	k	u
5	59	FK	U	f	m	v
6	0	QM	VW	gq	o	w

Example: Ndh0=QLiP-DwfL

Struktur des Netzwerks



- Segmentiert auf Netzebene und auf VLAN-Ebene, jedes VLAN eigenes Subnetz.
- Switch → nur in dein eigenes VLAN-LAN → Muss über den Router = Firewall
- Auf den Etagen kein Management VLAN.
  - ⇒ 3 Fehler nötig, damit Leute an den falschen VLAN rankommen.

Zusammenfassung von Vorlesung 5 (Koordination) I

-

Sicherheit

Kommunikation über unzuverlässige Infrastruktur.

Kernziel:

- Dde Wsqcbksqcbdle vml Iblel ulh Ibpek Tesk jeat ldcbt hsq Stpmkletz jsbk.
- Ibp sjtep Kulhelcbst ijedit veptpsujdcb, well Ibp Sepvep aeglscgt wdph.
- Sde epgellel mrrelqdcbtjdcbe Dsteljecgq ulh GeaelksßlsbkeI.
- Sde göllel edl Gebedkldq dl hep Gpunne surtedjel.

Vorweg: Absolut Sichere Systeme

1. Völlig vertrauliche Schnittstelle
2. Alles sofort vergessen

Vorweg: Absolut Sichere Systeme

1. Völlig vertrauliche Schnittstelle
2. Alles sofort vergessen

aber

1. nie völlig vertraulich
2. nutzlos — und schwierig

Vorweg 2: Schlüssellängen und Rechenzeit

- Länge → Entropie ⇒ $length \cdot log_2(N_{letters})$; bei echtem Zufall
  - Entropie 75: Bei schwachem Hash sicher bis etwa 2021[1]
  - Entropie 128: Laut ECRYPT bis 2028 für symmetrische Schlüssel, laut BSI bis 2022.[2]
  - Entropie 256: Laut ECRYPT bis 2068

Entropie 75 sind etwa 12 *zufällige* Zeichen oder 6 *zufällige* Wörter.

[1] Diskussion mit Beispielangriffen: draketo.de/english/secure-passwords
[2] Zusammenfassungen verschiedener Berichte: keylength.com

Server Sicher Aufsetzen

Damit Sie das vermeiden können:
„IT des Deutschen Bundestages fremdkontrolliert. Oppositionsabgeordnete ratlos." — https://www.draketo.de/it-des-bundestages-fremdkontrolliert-abgeordnete-ratlo

Informationen von **Markus Knye**, Leiter der IT bei Disy.

Vulnerability Management

- Alle Software-Versionen automatisiert gegen CVEs vergleichen (z.B. mit Nessus).

Literatur

Distributed Systems - An Algorithmic Approach – Sukumar Ghosh (2015).

Einzelziele:

- Sie kennen übliche Angriffe.
- Sie verstehen den Unterschied zwischen secret key und public key Kryptographie.
- Sie verstehen den Geburtstagsangriff auf Hashing-Algorithmen.
- Sie kennen die Auswirkung von Geburtstagsangriffen.
- Sie verstehen die Funktionsweise von PGP.
- Sie kennen Shamirs secret sharing.

Vorweg: Absolut Sichere Systeme

1. Völlig vertrauliche Schnittstelle
2. Alles sofort vergessen

aber

1. nie völlig vertraulich
2. nutzlos — und schwierig

Einwurf: Sichere Passwörter

- Passwortgeneratoren:
  `https://pthree.org/2018/04/19/use-a-good-password-generator`
- Diceware: `http://world.std.com/%7Ereinhold/diceware.html`
- `https://github.com/atoponce/webpassgen`
- `https://gist.github.com/atoponce/03109c0a51aededbddaf40b4c0aa0d7d`
- `https://www.draketo.de/software/letterblock-diceware`

*Weil Sie das wirklich **alle** kennen sollten.*

Grundlagen

- *Das* Richtig oder *Das* Falsch gibt es nicht.
- Installier möglichst wenig: Angriffsvektor klein halten. SSH absichern.
- Standardangriffsvektoren bei Nessus.
- Wisse, was du hast, und aktualisiere es. z.B. installier' Debian updates schnell.
- Verschlüsselung: Zero Trust ist ein buzzword: Sieh' sämtliche zugriffe als unsicher an ⇒ **Immer authentisieren**. Passwort, SSH Key. Prüf die SSL-Version!
  - Besorgnis bei Single Sign On: Bei Durchbruch ist jeder automatische Zugriff verloren.
  - Beispiel: MS Escher 2. Faktor, MS Authenticate. Haben angerufen. Kein Rate-Limit ⇒ nachts angerufen, bis Ja gedrückt.

Hardware

- Intel CPUs (Spectre) wären gefährlich, ist aber gefixt.
- Es gibt nur Intel für Blade-Server: Keine AMD-Blades. Lenovo bietet AMD Server an.
- ARM Blades gibt es, aber noch keine Erfahrung damit.

OS + Stack? Kernel? Hardening?

- Vanilla Debian. Härten den Kernel nicht zusätzlich, allerdings die Dienste (Apache darf z.B. nicht die Version nennen, SSH kein Root-Login per Passwort, . . .)
- SE-Linux lassen wir an, wie es ist. Muss manchmal wegen Problemen damit aus.

Windows-Server

- 2 Virenscanner: Einen Signaturbasierten, einen Verhaltensbasierten (gegen zero-days oder crypto-trojaner u.ä.).

Dienste / Software — Isolation/Container/Docker

- Microservices gegen heartbleed: Apache nur in einem Segment ⇒ Speicher auslesen macht nicht so viel.
- Segmentierung auch auf Service-Ebene.
- Docker: Nur ein Dienst pro Container.

Management-Schnittstelle

- IPXE boot, Standard-Installation automatisiert
  - Festplatten-Verschlüsselung,
  - SSH-Keys
  - . . .
  - binnen 15 min wieder genau gleich aufgesetzt.
  - Hilft bei Zurückverfolgung von Problemen: „was war der genaue Zustand vorher?“

Firewall

- 2 Firewalls
- Hardware Firewall: Appliance (Dedizierte Firewall) ⇒ Layer 3 Filter, nur Packet Filter, kann auch eine Iptables-Maschine sein. Nur eine Aufgabe.
  - Du musst 2 Fehler machen. Wenn du Port 80 ausversehen öffentlich bindest, musst du zusätzlich die Firewall(-s) fehlkonfigurieren.
- Blockt und loggt.

Logging / Monitoring / Intrusion Detection

- Logging und Analyse der Logs, auch auf der Firewall
- Elastic: Zentralisieren, Aggregieren, Dashboard-Filter, um den Unsinn rauszufiltern
  - Allein der Betrieb ist schwer, die Doku von Elastic ist *unvorteilhaft*
  - Alternativen: Greylog, . . .
  - . . . gut durchsuchen, filtern, . . .
    - du willst sehen, was jemand grade versucht.
    - Angriffe meist von Asiatischen, Russischen, Indischen IPs.
  - Ziemlich autonom, damit die Logs nicht so leicht korrumpiert werden können, wenn jemand durchbricht.
- Zur Sicherheit idealerweise ab Layer 2 monitoren, wäre aber unglaublich viel und Datenschutztechnisch schwierig

Monitoring: Datenschutz

- disy.net analysetools werfen die letzten beiden IP-blöcke weg (sind öffentliche Dienste ⇒ Datenschutz)
- unsere internen Dienste werden für 90 Tage voll geloggt. Sind nicht öffentlich (nur Mitarbeiter und Zustimmende) ⇒ dürfen wir. 90 Tage, weil nach mehr als 90 Tagen ein Angriff wahrscheinlich auch den Log-Server erwischt und die Logs korrumptiert hätte ⇒ wäre nutzlos.

Weiteres

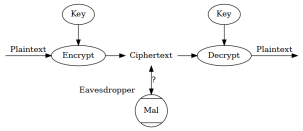
- BSI-Empfehlungen: https://www.bsi.bund.de/DE/Themen/Unternehmen-und-Organisationen/Informationen-und-Empfehlungen/Empfehlungen-nach-Angriffszielen/Server/server_node.html

Arten von Angriffen

- Lauschen (Eavesdropping)
- Überlasten (DoS: Denial-of-Service)
- Manipulieren (data tampering)
- Verstellen (masquerading — phishing)
- Mittelsmann-Angriff (MITM: man-in-the-middle)
- Malware installieren (Viren, Würmer, Spyware, Ransomware)

Verschlüsselung

- Klartext (plaintext)
- Verschlüsselt (ciphertext)
- Schlüssel (key)



Secret key - gemeinsames Geheimnis / symmetrische Schlüssel

- Schlüssel beiden Seiten bekannt
- Theoretisch sicherstes System: One-time pad
- Schlüsselaustausch?

Beispiel: Namens-Verschiebung

- abcd efghijklmnopqrstuvwxyz
- **cleo** abdfghijkmnpqrstuvwxyz

Vorgehensweise: Bekannte Sprachmuster finden.

Verbesserungen

Entropie erhöhen, um Muster auszulöschen

- Beispiel: Kompression.
- Ideal (Vergleich): Weißes Rauschen.
- Vorsicht: Kompressionsheader kann erkennbar sein.

⇒ Einfluss jedes Zeichens auf große Teile der Ausgabe. Konzeptuell ähnlich: Ganze Wörter austauschen.

Komplex und leicht zu verpatzen: Nutzen Sie existierende Bibliotheken!

Existierende Systeme

- DES: 56 bit key, heute zu schwach.
- AES: 128 bit blöcke, 128-256 bit key.
- One-time-pad: Sicher. Nächste Folie.

One-time-pad (Vernam cipher, bit-wise XOR)

- Pad: Bitstream
- Cipher: Klartext **xor** Pad
- Klartext: Cipher **xor** Pad

Jedes Pad darf nur einmal verwendet werden. Zufallsbits!

Analogie mit Verschiebung:

- Pad: DHBW = 4,8,2,23
- Daten: 24,23,14,-9

```
a b c d e f g h i j k l m n o p q r s t u v w x y z ä ö ü ß
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29
```

Problem: Austausch des Pad. Schlüssel so groß wie Daten!

Beispiel für One-time-pad

(An der Tafel)

Public key - Schlüsselaustausch umgehen

- Plaintext $P = D_d(E_e(P))$.

Rivest-Shamir-Adleman (RSA)

Anfang:

- s, t: Primzahlen
- $N = s \cdot t$
- Wähle d: keine gemeinsamen Teiler mit $(s-1) \cdot (t-1)$.
- Wähle e: $(e \cdot d) mod ((s-1) \cdot (t-1)) = 1$[3]

Verwendung:

- Cipher: $C = P^e mod N$
- Plaintext: $P = C^d mod N$

[3] d.h. $e \cdot d$ ist 1 mehr als ein Vielfaches von $(s-1) \cdot (t-1)$

Beispiel

$$
\begin{aligned}
\text{wähle zufällige } s, t &= 11, 13 && (1)\\
N &= 143 = 11 \cdot 13 && (2)\\
(s-1)\cdot(t-1) &= 10 \cdot 12 = 120 && (3)\\
\text{wähle: } d &= 7 && (4)\\
e &= 103\ ;\ (103 \cdot 7) mod 120 = 1 && (5)
\end{aligned}
$$

Öffentlich: $N = 143; e = 103$. Berechnung von d bräuchte s und t.

$$
\begin{aligned}
\text{Plaintext: } P &= 9 && (6)\\
\text{berechne } C &= 9^{103} mod 143 = 113 && (7)\\
\text{entschlüssle } P &= 113^{7} mod 143 = 9\ \checkmark && (8)
\end{aligned}
$$

Zusammenfassung

- Symmetrisch: Schlüssel beider Seiten.
- One-time-pad: C = P xor Pad
- Public key: $P = D_d(E_e(P))$
  - Veröffentlichen: e, N
  - Berechnung von d braucht $t \cdot s = N$

Hash - Inhalt Identifizieren oder Integrität prüfen

- Einfach: Quersumme
- Menschen: Name + Geburtstag + Wohnort
- Daten: SHA-256/SHA-512 (SHA-1 ist nicht mehr sicher)

Identifikation ≠ Integrität

Angriffe:

- Integrität: Zufällige Veränderung (z.B. Bitfehler)
- Identifikation: Anderer Inhalt (z.B. falsches Dateifragment)

Methoden:

- Integrität: CRC
- Identifikation: sha256

Praxis: Corona-Impfstatus

- Das geht natürlich mit Fümpf Blockchains![4]
- Oder mit ein paar Zeichen aus Name und Geburtsdatum.
  - Beschrieben ab 35:00 und nochmal aufgegriffen in Technisches Flair[5] ab 11:20 (ohne Fümpf Blockchains ☹)

[4] https://logbuch-netzpolitik.de/lnp385-fuempf-blockchains
[5] https://logbuch-netzpolitik.de/lnp386-technisches-flair

Hash-Identifikation brechen: Gewalt (brute force)

- Viele unterschiedliche Inhalte mappen auf den gleichen Hash.
- Einfach viele ausprobieren. Sehr viele.

Gegenmaßnahme zu Brute Force: Teure Hashfunktion

- Integrität ≠ Identität
- Integritätshash sollte schnell sein
- Identitätshash sollte langsam sein
- Möglichkeit: Mehrere Anwendungen
- Beispiel: **bcrypt**

Geburtstagsangriff

Geburtstage sammeln?

alle, die sie schreiben wollen, in neuem Cryptpad

Geburtstagsangriff

Geburtstage sammeln?

alle, die sie schreiben wollen, in neuem Cryptpad

- N Personen
- $N-1$ Vergleiche pro Person
- $N(N-1)$ Vergleiche.
- $27 \cdot 26 = 702$
- Kollisionswahrscheinlichkeit bei 2 Personen: $\frac{1}{365} \approx 0.27\%$
- **Keine** Kollision bei 702 Vergleichen: $(1 - \frac{1}{365})^{702} \approx 14.6\%$
- **85% Wahrscheinlichkeit einer Kollision**

Geburtstagsangriff gegen Hashes

- Bedrohung: Passwort als Hash gespeichert. Angreifer hat die Datenbank.
- Problem: Braucht nur irgendeinen Treffer → Vergleiche mit allen Hashes.
- Gegenmaßnahme: Viel, viel längere Geheimnisse.

Hash-Vorberechnen: Rainbow-tables

- Alle plausiblen Passwörter bis 8 Zeichen vorhashen
- Hashes speichern ⇒ map: Hash → Passwort
- Hash-once-crack-everything

Gegen Rainbow tables: Salt

Für jeden Nutzer einen anderen zufälligen key mithashen; einfach mit Hash speichern, z.B. in `/etc/shadow`.

```
$Algo$Salt$Hash...
```

Beispiel:[6]

```
aychedee:$6$vb1t...$M.1ZCqKtJ.../qNxLyqW.U/:15405:0:99999:7:::
```

[6] https://www.aychedee.com/2012/03/14/etc_shadow-password-hash-formats/

Signaturen

- Integrität garantieren
- Autorenschaft bestätigen

Integrität: Hash des Inhalts.

Autorenschaft: Signieren. Inverser öffentlicher Schlüssel!

Beispielsignatur

- Inhalt: 135 → Quersumme 9
- Öffentlich: $e = 103, N = 148$
- Privat: $s' = d = 7$

$$
\begin{aligned}
P &= 9 && (9)\\
C &= 9^{7} mod 143 = 48 && (10)
\end{aligned}
$$

Nachricht: **135,48**

Beispielsignatur prüfen

Nachricht: **135,48**

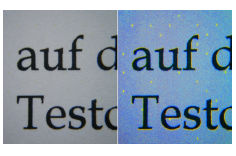
- Berechnet: Quersumme von 135 ist 9
- Öffentlich: $e = 103, N = 148$

$$
\begin{aligned}
C &= 48 && (11)\\
P &= 48^{103} mod 143 = 9\ \checkmark && (12)
\end{aligned}
$$

Die mit dem privaten Schlüssel signierte Quersumme passt zum Dokument. Es gibt nur **ein** passendes C kleiner 143.

Steganographie, Beispiel

Verstecken, dass überhaupt Daten ausgetauscht werden.



Seriennummer, Datum und Uhrzeit, Daten zur Fehlerkorrektur.
Bild von Florian Heise, public domain, https://commons.wikimedia.org/wiki/File:HP_Color_Laserjet_3700_schutz_g.jpg

PGP - praktische Verschlüsselung

GnuPG is a complete and free implementation of the OpenPGP standard as defined by RFC4880 (also known as PGP)
— https://gnupg.org/

- Signieren
- Verschlüsseln

Verschlüsseln mit public key

```
echo Hello World > example.txt
gpg --armor --encrypt --recipient arne_bab@web.de example.txt
cat example.txt.asc

-----BEGIN PGP MESSAGE-----

hQIMA59FFTDQ4LRMAQ/+Ppx8FpxodKEkKu9+KyodnOhE5r9jKV0OcvHx4gEIYUuU
wTlbYIwkfIfqeOyqnAOAN+kDcSFLLX86L1vL6E16FmBB/ZL9xdxQgVJ1sXUyCOJz
NhIFfplNyqIMvDgOcEH2c8fSHQQz1XRS8XFpfc/Ffk2OgKB2EfqwAUa+3oLLlvZI
7BBQBMrTnvrdefiEqGR36pARBLjDHFV8dk/L7Ak3Xp85mPydr9c6GnmCtm2AO03x
...
saGEw/0H5hN2gZo5EMbWO+D+NowwXQ3hlz3G4zrdGag2bur4a5sYzm3Bm48iOSLF
fLqOS7pFPF4siUd71pIiLlvu9xK7Z2Nkj6POrNszJB9uEw==
=rPXK
-----END PGP MESSAGE-----
```

Entschlüsseln

```
gpg -o /tmp/example.txt --decrypt example.txt.asc
cat /tmp/example.txt

Hello World
```


Signieren

```
echo Hello World > example-sign.txt
gpg --armor -b --sign example-sign.txt
cat example-sign.txt.asc

-----BEGIN PGP MESSAGE-----

owNCWmg5MUFZJlNZua2qrgAAuX/////7v33/n9u+/+/d////b/X33//59737///b
//v/v/bAAb3d21mukBoBoAOAyAAAOOGgHqAAAAAAANAOAAAOAAAAAaGgNGEAOANB
...
OGSiJjn1BCCH9jIKdkZTqdmZejDQhLojkRn2PW4EAB8j/F3JFOFCQua2qrg=
=UzZn
-----END PGP MESSAGE-----
```

(auch abgetrennt möglich)

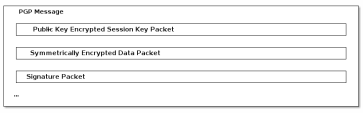
Signatur prüfen

```
echo Hello World > example-sign.txt
gpg --verify example-sign.txt.asc 2>&1

gpg: Signatur vom Mo 15 Apr 2019 23:24:02 CEST
gpg: [...] mittels RSA-Schlüssel F34D6A1235D04903CD22D5C013EF8D452403C3EB
gpg: Korrekte Signatur von "Arne Babenhauserheide (Drak) <arne_bab@web.de>" [ul
gpg: WARNUNG: Keine abgetrennte Signatur; die Datei 'example-sign.txt' wurde NI
```


Funktionsweise

- Verschlüsselt Inhalt mit symmetrischem Schlüssel (schnell!)
- Verschlüsselt symmetrischen Schlüssel mit öffentlichen Schlüsseln
- Größe der Mail ≈ O(1) mit der Zahl der Empfänger



→ https://www.ietf.org/rfc/rfc4880.txt

Verbesserungen für E-Mails

- pEp: Pretty Easy Privacy: https://prettyeasyprivacy.com/
  - Verschlüsselt Betreff und andere Header (sind im Body!).
  - Vergleich: Cypherpunk remailer → https://en.wikipedia.org/wiki/Cypherpunk_anonymous_remailer
- Autocrypt: https://autocrypt.org/
  - Schlüssel via Header mitschicken: Sender und Empfänger
  - Peer-state Header
  - Opportunistisch ≈ Trust-on-first-use (Tofu)
    - Aber als zu schwach kritisiert: Gegenseite kann Schlüssel austauschen https://en.wikipedia.org/w/index.php?title=Autocrypt&oldid=1009178647#Security_model

Außerdem: Einfachere Bibliothek: https://sequoia-pgp.org/
— GnuPG-Key auf Papier:
https://www.jabberwocky.com/software/paperkey/

Header im Cypherpunk remailer

```
::
Anon-To: <Recipient Email Address>

##
Subject: <Subject>

<Message Text>

Verschlüsseln ⇒

::
Encrypted: PGP

-----BEGIN PGP MESSAGE-----
<place encrypted output here>
-----END PGP MESSAGE-----
```


Autocrypt-Beispiel

```
Delivered-To: <bob@autocrypt.example>
From: Alice <alice@autocrypt.example>
To: Bob <bob@autocrypt.example>
Subject: an Autocrypt header example using Ed25519+Cv25519 key
Autocrypt: addr=alice@autocrypt.example; prefer-encrypt=mutual; keydata=
 mDMEXEcE6RYJKwYBBAHaRw8BAQdArjWwk3FAqyiFbFBKT4TzXcVBqPTB3gmzlC/Ub7O1u120F2F
 saWNlQGFidG9jcnlwdC5leGFtcGxlIJYEExYIAD4WIQTrhbtfozp14V6UTmPyMVUMT0fjjgUCXE
 cE6QIbAwUJA8JnAAULCQgHAgYVCgkICwIEFgIDAQIeAQIXgAAKCRDyMVUMT0fjjkqLAP9frlijw
 BJvA+HPnqC2cYIVxlyKz55Ci5gWTpp37K73jgD/VbKYhkwk91u6890YH4K7q7LbmdeaJ+RX88Y/
 ad9bZwy4OARcRwTpEgorBgEEAZdVAQUBAQdAQv8GIa2rST2zgqbXCpDDYMiKRVitCsy203x3sES+
 eviIDAQgHiHgEGBYIACAWIQTrhbtfozp14V6UTmPyMVUMT0fjjgUCXEcE6QIbDAAKCRDyMVUMT0
 fjjlnQAQDFHUs6TIcxrNTtZFjUFm1M0PJ1Dng/cDW4xN8OfsmOQEA22Kr7VkCjeAEC08VSTeV+
 QFsmz55/lntWkwYWhmvOgE=
Date: Tue, 22 Jan 2019 12:56:25 +0100
MIME-Version: 1.0
Content-Type: text/plain

This is an example e-mail with Autocrypt header
as defined in Level 1 revision 1.1.
```


SSL/TLS - verbreitete Verschlüsselung

- Zertifikate via Hierarchie ⇒ Konfiguration im Client mitgeliefert
- Handshake mit public-key → symmetrischer Schlüssel für Daten
- Auch für Java RMI, aber hässlich — mit webstart verdammt hässlich. Webstart ist tot. Fast tot. Untot.

SSL Zertifikat

- Selbstsigniert möglich, aber mit grässlicher Warnung
- Certificate authority → teuer, umständlich
- Automatisiert: letsencrypt → https://letsencrypt.org/

Shamirs Secret Sharing

Vingt ans après, Autor Unbekannt, 1864, Publisher: J.-B. Fellens et L.-P. Dufour → https://commons.wikimedia.org/wiki/File:Dumas_-_Vingt_ans_apr%C3%A8s,_1846,_figure_page_0240.png

Shamir's Secret Sharing Scheme: split

```
./ssss-split -t 2 -n 4

Generating shares using a (2,4) scheme
      with dynamic security level.
Enter the secret, at most 128 ASCII characters: ...

      Using a 80 bit security level.
1-a419df48569dc3aeec35
2-8a73455bbc70d5c3246a
3-6faaccaae5d427e79dae
4-d6a6717c69aaf918b4ca
```


Shamir's Secret Sharing Scheme: combine

```
./ssss-combine -t 2

Enter 2 shares separated by newlines:
Share [1/2]: 1-a419df48569dc3aeec35
Share [2/2]: 3-6faaccaae5d427e79dae
Resulting secret: D'Artagnan
```

- Werkzeug: http://point-at-infinity.org/ssss/
- Methode: en.wikipedia.org/wiki/Polynomial_interpolation

Zusammenfassung

- **Passwortgenerator**! Oder diceware.
- **Angriffe** von Lauschen bis Überlast (DoS) und Verstellen. MITM und Malware.
- **Symmetrische** Verschlüsselung → gleicher Schlüssel. Billiger.
- **Public-key**: Öffentlicher Teil eines Schlüsselpaares. Löst Schlüsselaustausch.
- **Hash** Identifiziert oder sichert Integrität.
  - **Geburtstagsangriff**: 85% Kollisions-Wahrscheinlichkeit hier im Raum.
  - **Salzen** Sie Ihre Passworthashes.
- **PGP und SSL**: Public key → symmetric session key.
- **Shamir's Secret Sharing**: Brauchen k von N zur Rekonstruktion.

Fragen für die Prüfung?

Ideensammlung hier in der Vorlesung an der Tafel:

-
-
-
-

Viel Erfolg auf Ihrem weiteren Weg!

Und denkt an die Losung von Früchte des Zorns:
„Wir passen aufeinander auf“

Verweise I

Ghosh, S. (2015). *Distributed Systems - An Algorithmic Approach*. Computer & Information Science. Chapman & Hall/CRC, 2 edition, ISBN: 978-1466552975.