Willkommen bei Verteilte Systeme!

Von Datenbanken
über Webdienste
bis zu p2p und Sensornetzen.

Heute: **Sensornetze und Sicherheit**.

Zusammenfassung von Vorlesung 5 (Koordination) I

- Koordinator vereinfacht Algorithmen
- Synchronisierer ermöglichen synchrone Algorithmen in asynchronen Systemen
- Fehler: Crash, Auslassung, Byzantinisch
- Toleranz: Maskierend?
- Erkennung: Vollständigkeit, Korrektheit
- Selbststabilisierung
- Konsens: Byzantinische Generäle
- Sensornetze: Energie, Selbstorganisation, Sicherheit

Literatur

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

Ablauf heute

- Sensornetze
- Sicherheit

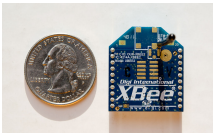
Sensornetze

Kommunizierende, selbstorganisierende Minirechner.

Ziele:

- Sie kennen die zentralen Herausforderungen für Sensornetze.

Mote, Beispiel



XBee Series 2 with Whip Antenna, with US Quarter.jpg, Mark Fickett CC BY 3.0, wikimedia.

Daten (XBee ZigBee (S2C)):

- Stromverbrauch: 1μA (schlafend) bis 59mA (sendend)
- Drahtlose Bandbreite: 250kbit/s

Herausforderungen

- Energie sparen: Jahre mit Batterie
- Fehlertoleranz: Ausfall vieler Knoten erwartet
- Selbstorganisation (Kommunikation und Organisation, mobil)
- Zeitsynchronisierung (nach Schlaf!)
- Sicherheit: Angreifer haben mehr Energie!

Energie

- Faktor 50000 zwischen Verbrauch bei Aktivität und Schlaf!
- Algorithmen optimieren
- Kommunikation ist teuer
- Daten sammeln, zusammenfassen, von gewähltem Knoten weiterleiten lassen
- Sender durchwechseln

Selbstorganisation

- Ausbringung ohne Setup → Messgeräte in Waldgebiet
- Daten weiterleiten → größere Reichweite
- Beweglich: Optimierung der Position
- Mit oder ohne Basisstation

Sicherheit

- Ein Laptop hat mehr Energie als das gesamte Netzwerk
- neue Bedrohungsszenarien → Mechanismen, die mit wenig lokaler Leistung auskommen

Zusammenfassung Sensornetze

Zentrale Herausforderungen:

- Energie
- Fehlertoleranz
- Selbstorganisation
- Zeit
- Sicherheit

Ziele

- 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

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: 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

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

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.*

Beispiel: Letterblock Diceware Vorderseite

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=0LiP-Dwfl

Beispiel: Letterblock Diceware Rückseite

Letterblock Diceware

Roll **two dice per letter** (see other side). If you rolled two letters (e.g. BC), choose one at will. Roll at least **two blocks of four letters**. Each block has ≈20 bits entropy. For block **separators** add all rolled row-numbers of two consecutive blocks and take modulo 6:

0	1	2	3	4	5
.	+	-	=	@	%

Zusammenfassung

- nie völlig vertraulich
- nutzlos — und schwierig
- Sichere Passwörter!

Server Sicher Aufsetzen

Damit Sie das vermeiden können:

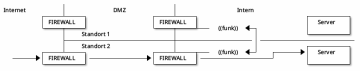
„IT des Deutschen Bundestages fremdkontrolliert. Oppositionsabgeordnete ratlos.“ — https://www.draketo.de/it-des-bundestages-fremdkontrolliert-abgeordnete-ratlo

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

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.

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.

Vulnerability Management

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

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

Signaturen

- Integrität garantieren
- Autorenschaft bestätigen

Integrität: Hash des Inhalts.

Autorenschaft: Signieren. Inverser öffentlicher Schlüssel!

Beispielsignatur

- Schlüssel wie gehabt.
- Inhalt: 135 → Quersumme 9
- Öffentlich: $e = 103, N = 143$
- Privat: $s' = d = 7$

$$P = 9 \quad (1)$$
$$C = 9^7 mod 143 = 48 \quad (2)$$

Nachricht: **135,48**

Beispielsignatur prüfen

Nachricht: **135,48**

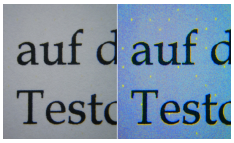
- Berechnet: Quersumme von 135 ist 9
- Öffentlich: $e = 103, N = 143$

$$C = 48 \quad (3)$$
$$P = 48^{103} mod 143 = 9 \checkmark \quad (4)$$

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, commons.wikimedia.org/wiki/File:HP_Color_Laserjet_3700_schutz_g.jpg

PGP - praktische Verschlüsselung

GnuPG is a complete and free implementation of the Open-PGP 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+KyodnUhE5r9jKV0OcvHx4gEIYUuU
wTlbYIwkfIfqeQyqnAOAN+kDcSFLLX86L1wL6El6FmBB/ZL9xdxQgVJlsXUyCOJz
NhIFfplNyqIMvDgOcEH2c8fSHQQz1XRS8XFpfc/FfkZOgKB2EfqwAUa+3oLLlv2I
7BBQBMrTnvrdefiEqGR36pABBLjDHFV8dk/L7Ak3Xp85mPydr9c6GnmCtm2A003x
...
saGEw/OH5hN2gZo5EMbW0+D+NowwXQ3hlz3G4zrdGag2bur4a5sYzm3Bm48iOSLF
fLqO57pFPF4siUd71pIiLlvu9xK7Z2Nkj6PGrNszJB9uEw==
=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/bAAb3d21mukBoBoAOAyAAA0OGgHqAAAAAAANAOAAAOAAAAAaGgNGEAOANB
...
CGSiJjn1BCCH9jIKdkZTqdmZejDQhLojkRn2PW4EAB8j/F3JFOFCQua2qrg=
=UzZn
-----END PGP MESSAGE-----
```

(auch abgetrennt möglich)

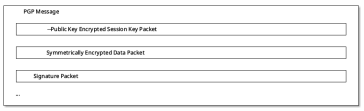
Signatur prüfen

```
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

- 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

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
 saWN1QGF1dG9jcnlwdC5leGFtcGxliJYEExYIAD4WIQTrhbtfozp14V6UTmPyMVUMT0fjjgUCXE
 cE6QIbAwUJA8JnAAULCQgHAgYVCgkICwIEFgIDAQIeAQIXgAAKCRDyMVUMT0fjjkqLAP9frlijw
 BJvA+HFnqCZcYIVx1yXzSSGi5gMTpp37K73jgD/VbKYhkwk9iu689OYH4K7q7LbmdeaJ+RX88Y/
 ad9hZwy4OARcRwTpEgorBgEEAZdVAQUBAQdAQv8GIa2rSTzgqbXCpDDYMiKRVitCsy203x3sE9+
 eviIDAQgHiHgEGBYIACAWIQTrhbtfozp14V6UTmPyMVUMT0fjjgUCXEcE6QIbDAAKCRDyMVUMT0
 fjjlnQAQDFHUs6TIcxrNTtZFjUFm1MOPJ1Dng/cDW4xN8OfsnOQEA22Kr7VkCjeAEC08VSTeV+
 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.
```


- Vergleich: Tor
- Aber: kaum Nutzung: 90% von 1 Person
  - ⇒ busted
  - ⇒ Teach or be caught

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 → 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

Ungelöst: wie prüfe ich, ob splits korrekt sind, ohne Zugriff auf die Originaldaten zu haben?

Zusammenfassung

- **Sensornetze**: Energie, Selbstorganisation, Sicherheit
- **Passwortgenerator**! Oder diceware.
- **PGP und SSL**: Public key → symmetric session key.
- **Shamir's Secret Sharing**: Brauchen k von N zur Rekonstruktion.

Viel Erfolg auf Ihrem weiteren Weg!

Und denkt an die Losung von Früchte des Zorns:

„Wir passen aufeinander auf“

Und falls Ihr Interesse an Arbeit mit Disy habt oder andere kennt, die es haben, kontaktiert mich gerne: arne.babenhauserheide@disy.net oder karriere@disy.net (z.B. zu Master-Möglichkeiten)

Verweise I

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