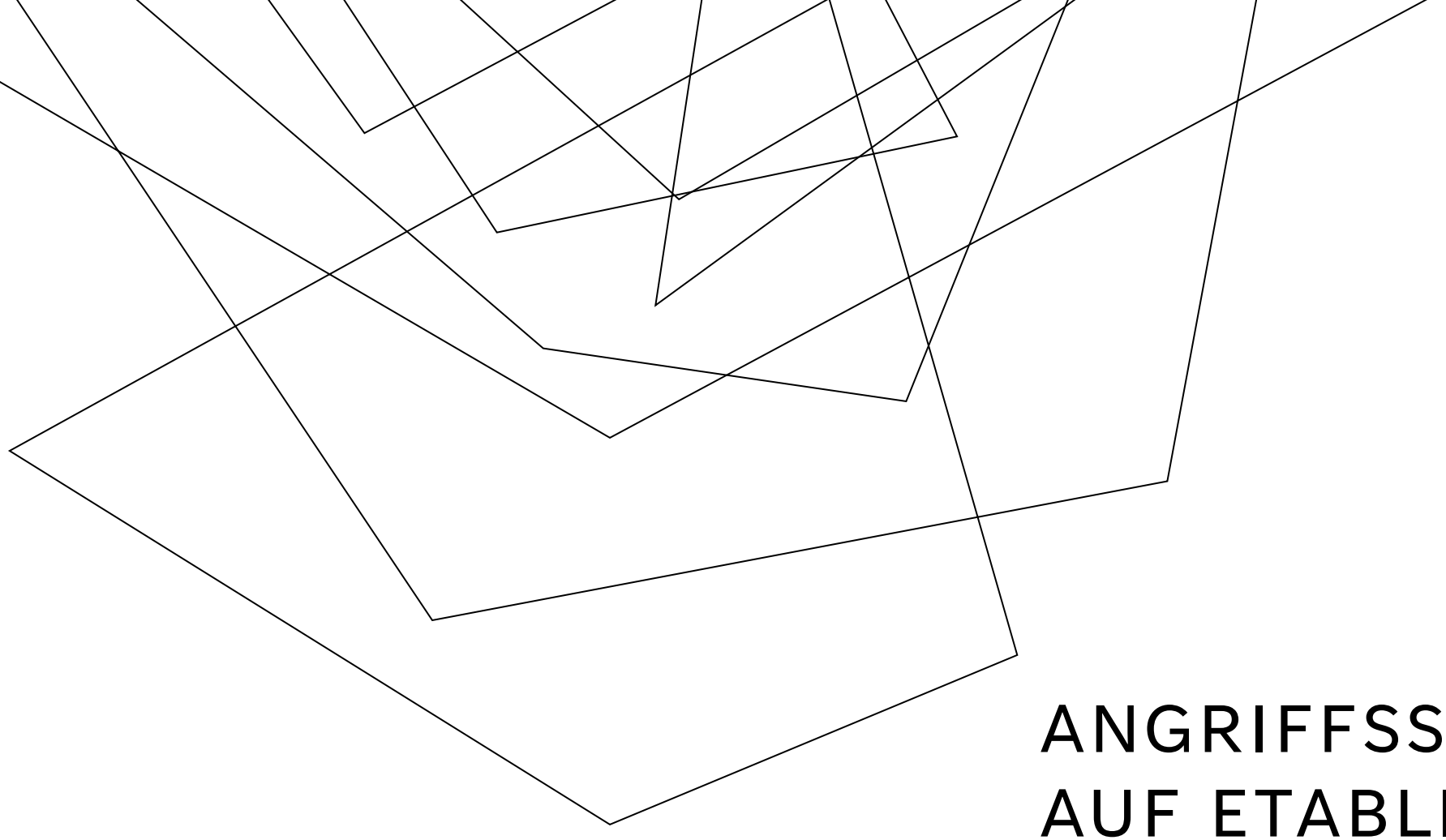


ANGRIFFSSZENARIEN AUF ETABLIERTE NETZWERKPROTOKOLLE

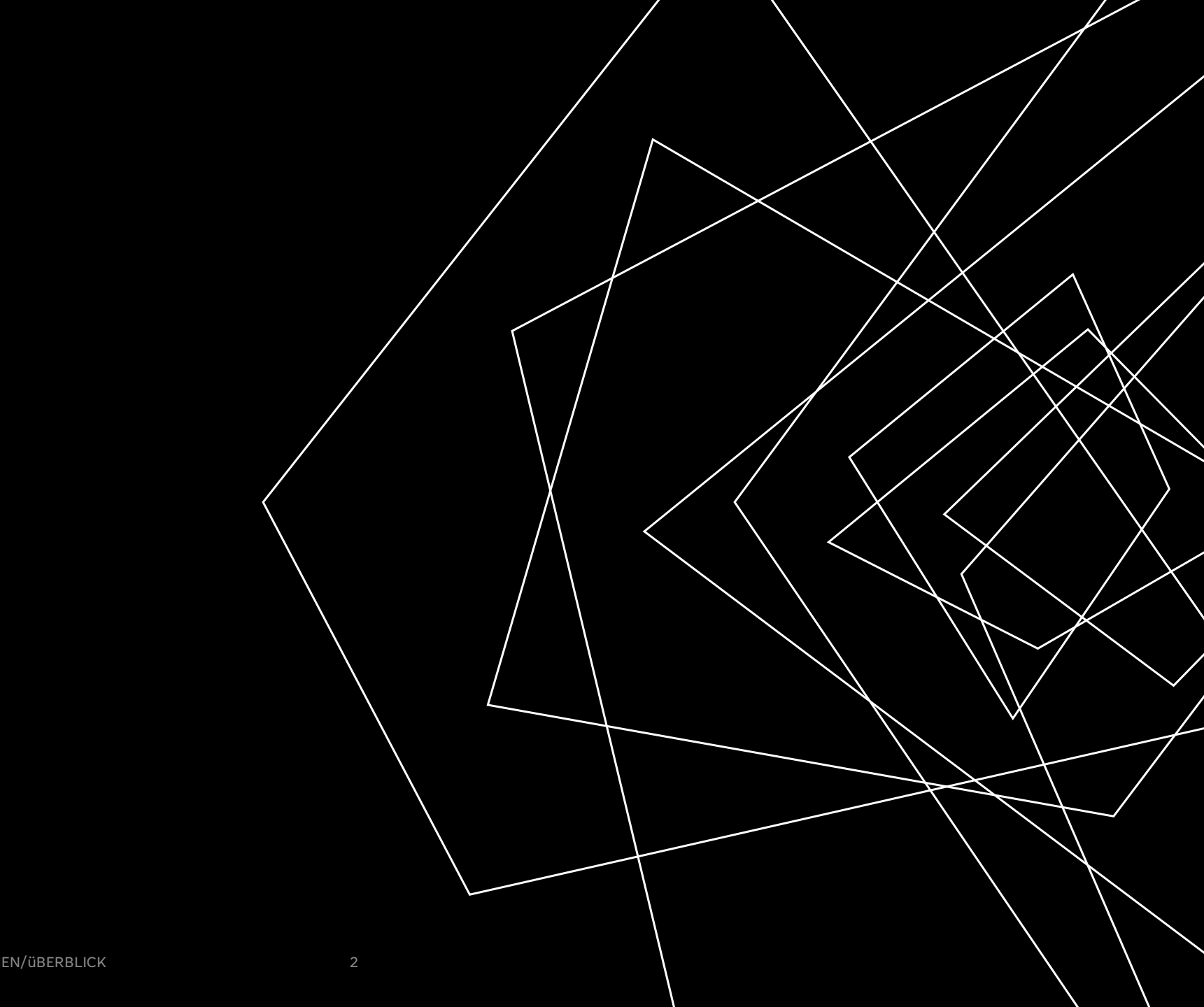
Kaan Kaygısız

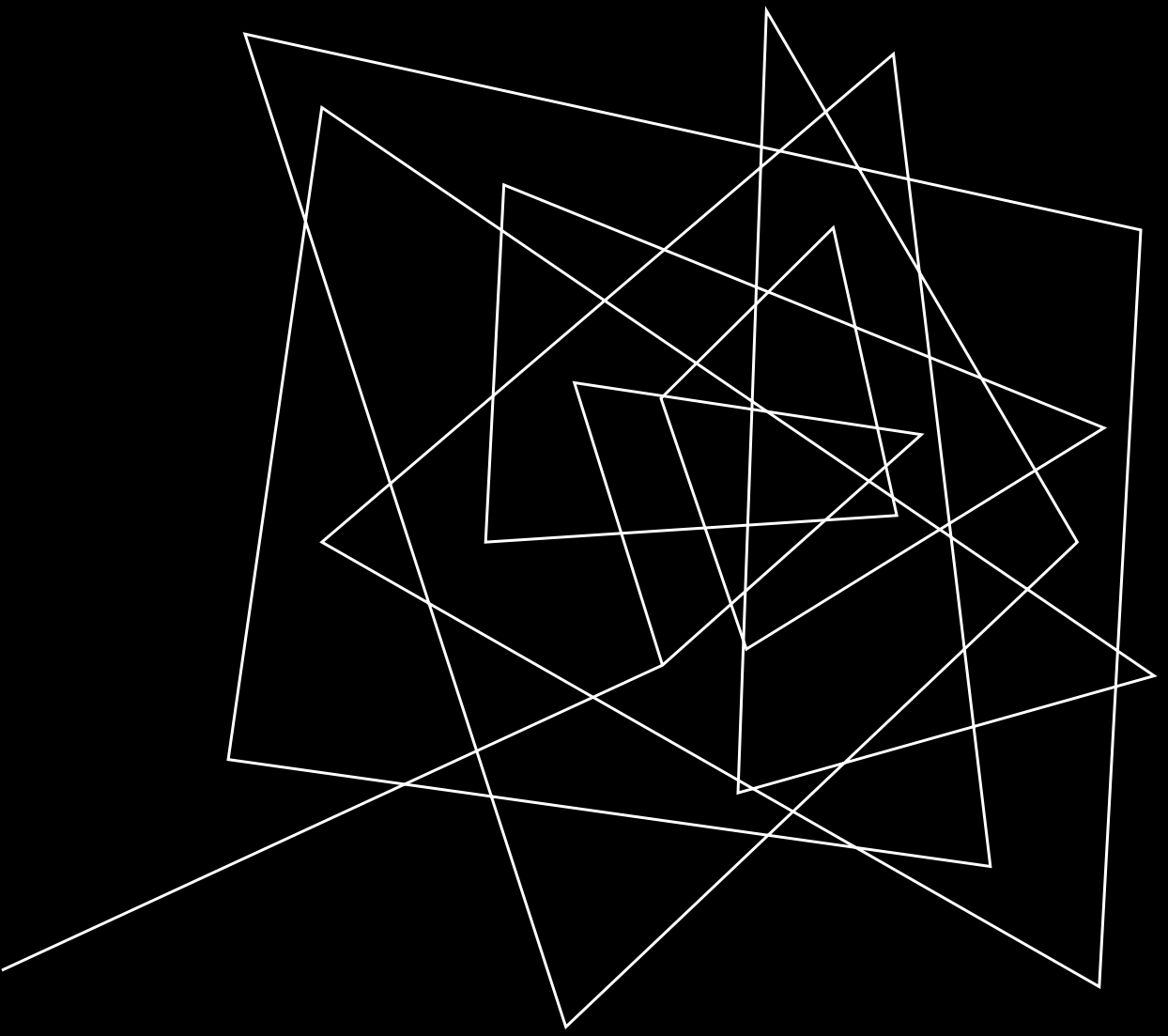
ÜBERBLICK

I – WannaCry

2 – Heartbleed

3 – Prävention





ETERNALBLUE & WANNACRY

Kapitel I der Seminararbeit

ETERNALBLUE

- Schwachstelle in der **Microsoft-Implementation** von **SMB** (= 'Server Message Block')
- **Ermöglicht** Angreifern **zugriff** auf das System **mit *System-Rechten***

WANNACRY

- Ransomworm, der die EternalBlue-Schwachstelle ausnutzt
- Target: u.a. ältere Versionen von Microsoft Windows XP, Vista, 7 und 10.
- Verschlüsselt System-Daten mit *AES (→Hybride Verschlüsselung)*
- Verlangt Geld, um den Schlüssel freizugeben
- U.A. das BSI und Microsoft raten ***gegen*** die Zahlung des Lösegelds.

SCHLÜSSELUMGANG

Schritt I: Infektion

System wird infiziert:

→Generierung eines Schlüsselpaars & eines AES-Schlüssels

Schritt II: AES

System wird verschlüsselt:

→ Alle Dateien werden mit AES verschlüsselt

→ AES ist ein schneller Algorithmus

→ System ist in kurzer Zeit vollkommen verschlüsselt

Schritt III: Schlüsseln

Die Schlüssel werden verschlüsselt:

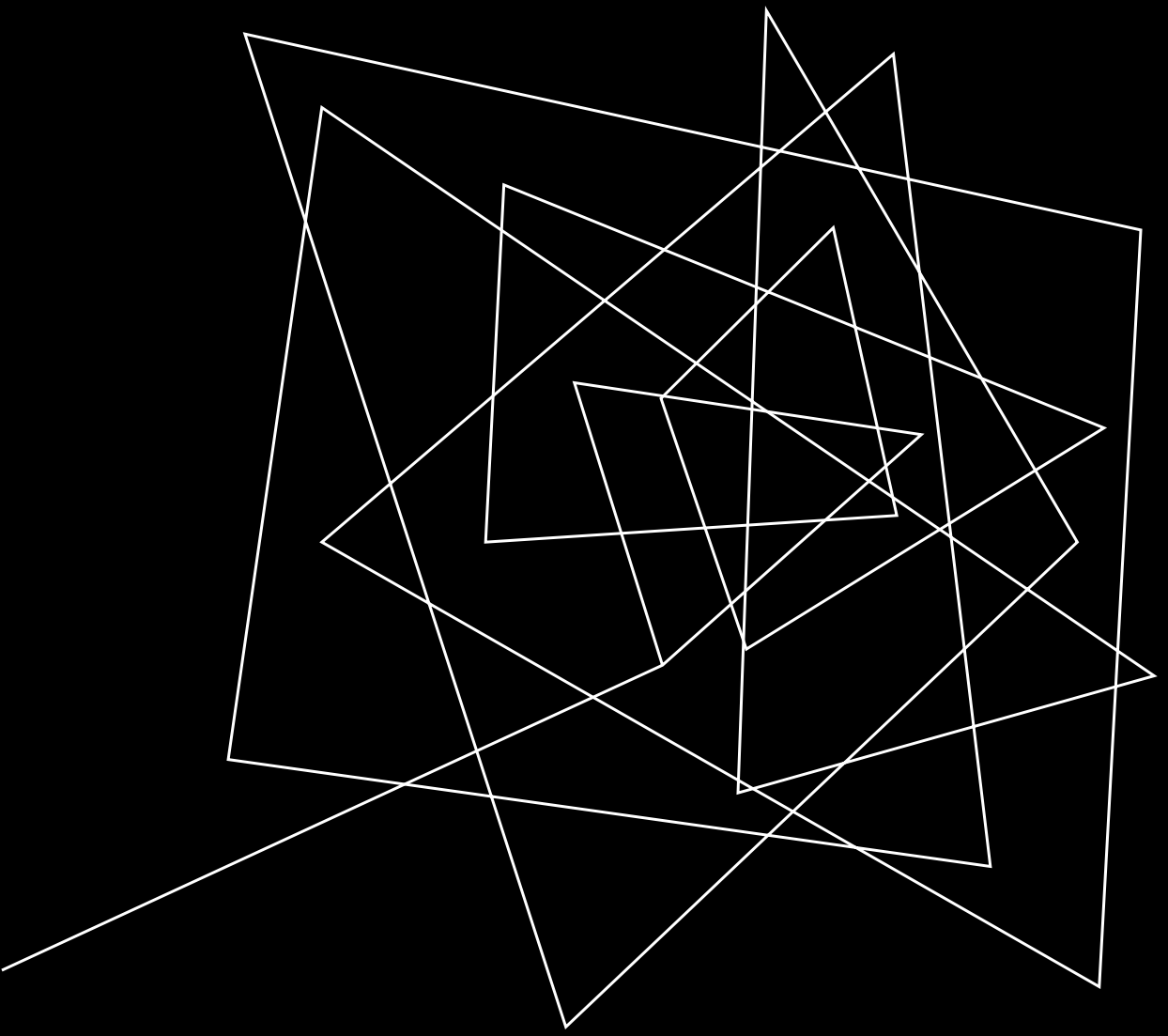
→ Der AES-Schlüssel wird mit dem generierten *public key* verschlüsselt.

→ Der *private key* wird mit dem öffentlichen key des Kontrollservers verschlüsselt.

Schritt IV: Versenden

Der verschlüsselte *private key* wird dem Kontrollserver gesendet

*...und **hoffentlich** nach Zahlung entschlüsselt.*



OPENSSL & HEART*BLEED*

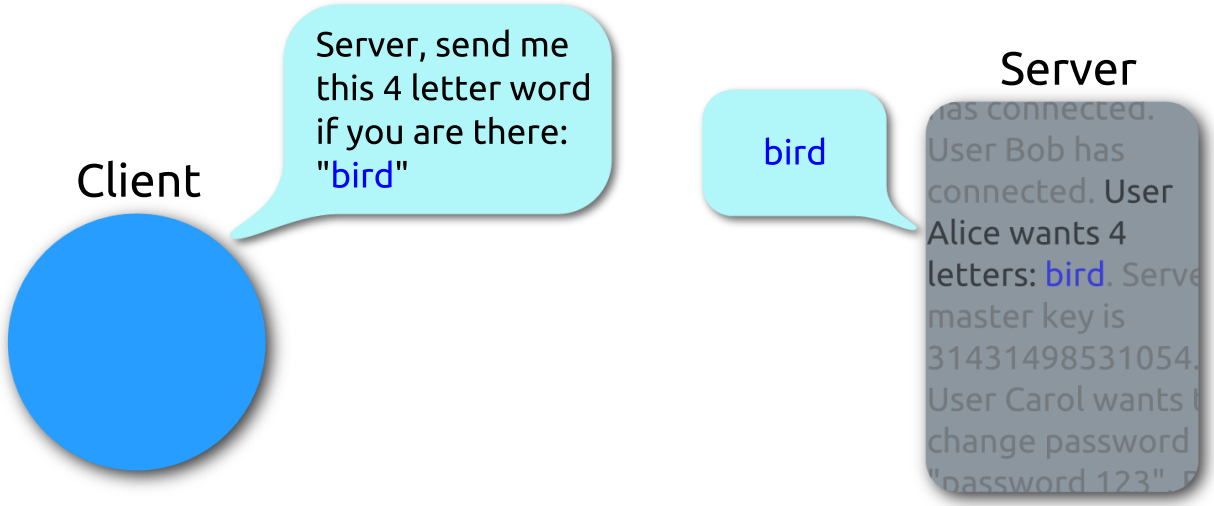
Kapitel 2.2 der Seminararbeit

OPENSSL: WAS IST DAS?

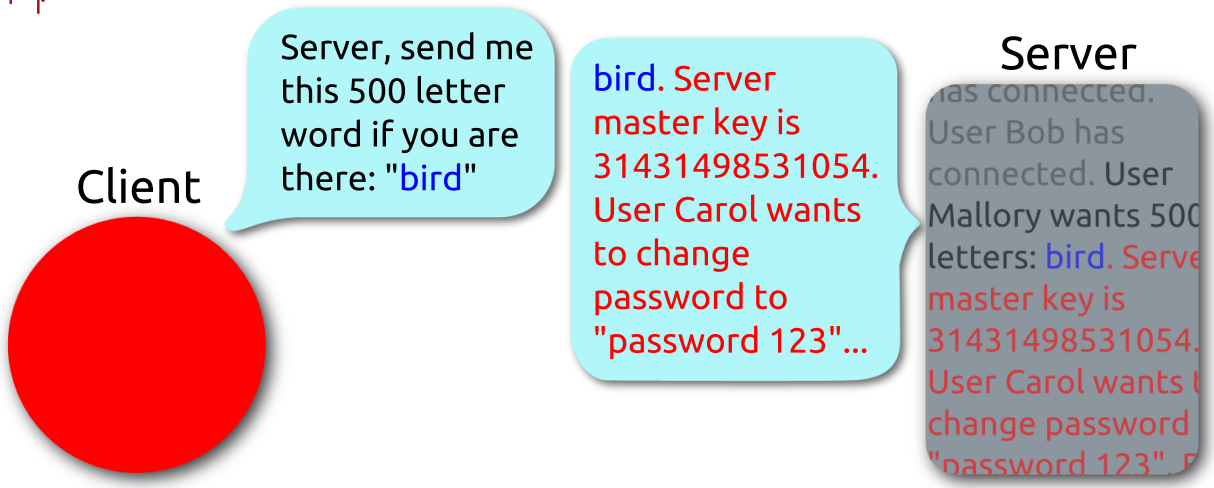
- Open-Source C/C++ Bibliothek für die einfachere implementation von TLS bzw. SSL (→ HTTPS)
- Sehr weit verbreitet, da es einige Funktionen bietet
- U.a. zwei der am weitesten verbreiteten Engines (Apache HTTPD & NGINX) nutzen OpenSSL*

**wenn das SSL modul eingeschaltet ist*

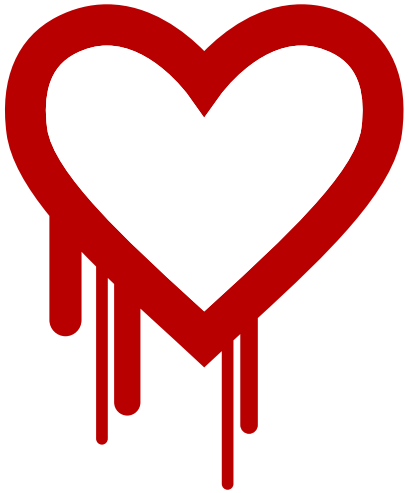
Heartbeat – Normal usage

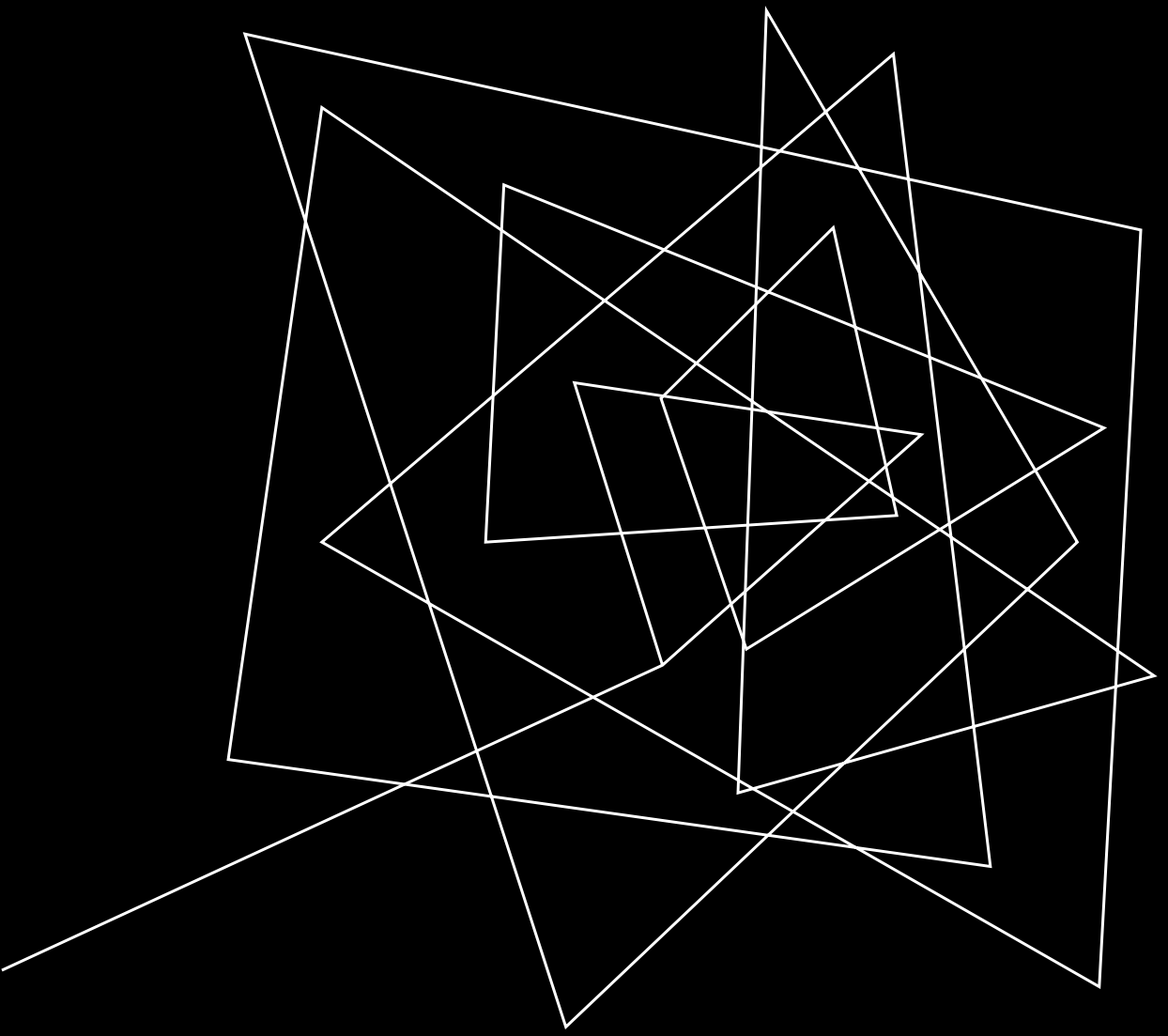


Heartbeat – Malicious usage



HEARTBLEED





PRÄVENTION: *END-NUTZER*

Kapitel 3.1 der Seminararbeit

EMPFEHLUNGEN

→ Installation nur aus vertrauten Quellen

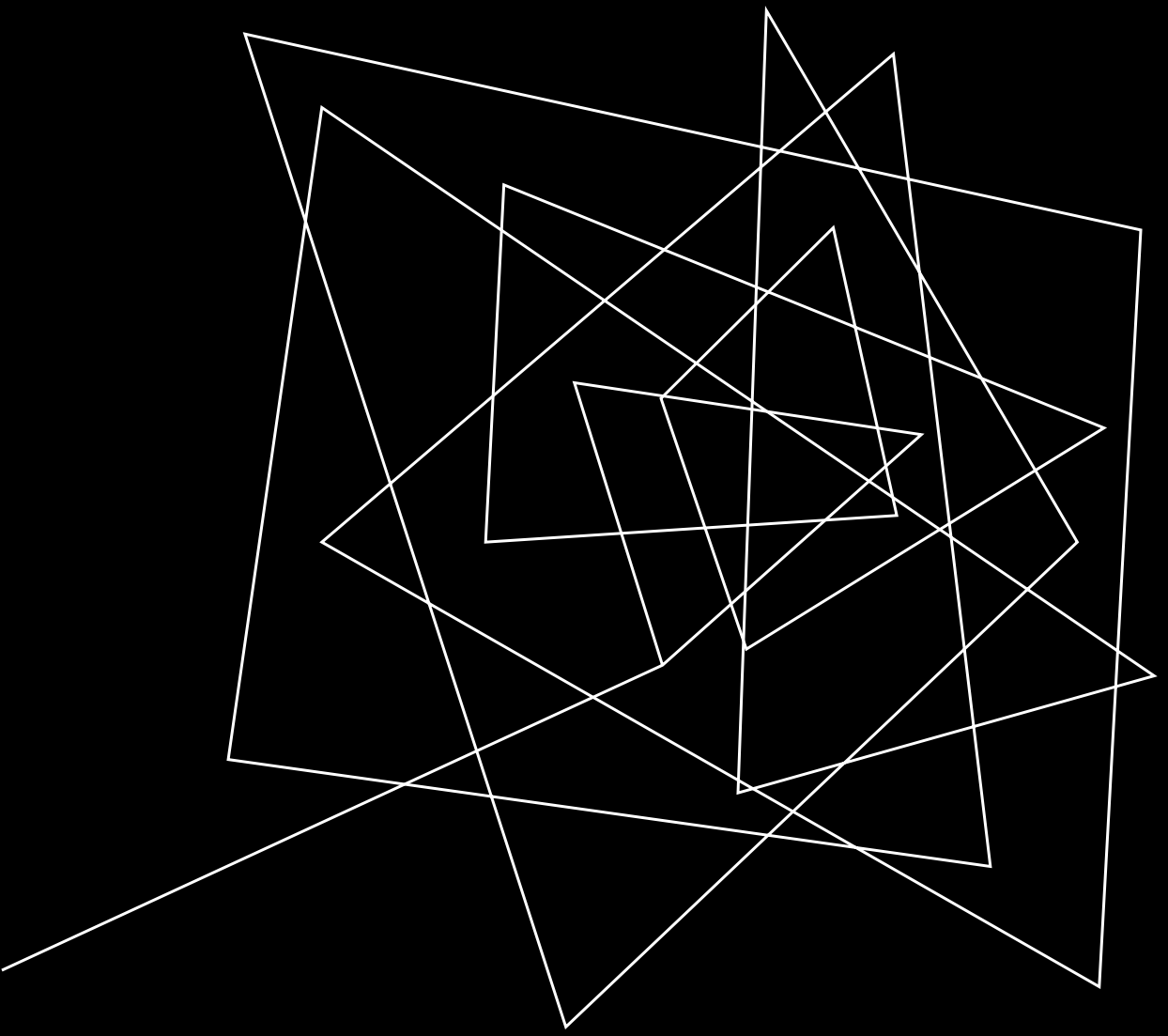
→ System und Apps immer *up-to-date* halten

→ Vor der Weitergabe von Information: Ist die URL richtig?

→ Auf FOSS/OSS(= (Free &) Open-Source Software) setzen[1]

Aber: auch OSS ist nicht unfehlbar und hat Fehler!

(→ s. Kapitel "*OpenSSL & Heartbleed*")



PRÄVENTION: *ENTWICKLER*

Kapitel 3.2 der Seminararbeit

EMPFEHLUNGEN

→ Nutzen von Kommentaren (→ JavaDocs [für Java 8])

→ Versions-kontrollsysteme (Git)

→ *Rubber Duck Debugging*

→ Halte dich an die Standards (**Coding Conventions [Java]**)

RUBBER DUCK DEBUGGING

- Beg, borrow, steal, buy, fabricate or otherwise **obtain a rubber duck** (bathtub variety).
- **Place rubber duck on desk** and *inform it you are just going to go over some code with it, if that's all right*.
- **Explain** to the duck **what your code is supposed to do**, and **then go into detail** and **explain your code line by line**.
- **At some point** you will tell the duck **what you are [supposed to be] doing next** and then **realise that that is not in fact what you are actually doing**. The duck will sit there serenely, happy in the knowledge that it has helped you on your way.

DANKE FÜR EURE AUFMERKSAMKEIT

Quellen, Handout und Seminararbeit unter

github.com/akb1154/seminararbeit