Orsum occulendi

AES in Java | PKN

Lukas Bischof, Philipp Fehr

Inhaltsverzeichnis

Einleitung

Philipp Fehr und Lukas Bischof haben in dem Projekt „Orsum Occulendi“ versucht, den AES-Algorithmus in Java umzusetzen. Damit die Kommunikation in unserem bereits vorhandenen Java Vier-Gewinnt verschlüsselt werden kann.
Während der Entwicklung wurde das Projekt zusätzlich von Herrn Tromsdorff und Herrn Veselcic betreut.

Das Problem

Das zentrale Problem des Projekt ist, dass die Kommunikation unseres Java Vier-Gewinnt Spieles nicht verschlüsselt übertragen wird und somit eine erhebliche Sicherheitslücke aufweist, da man mit einer Man-In-The-Middle Attacke die Spiele verändert sowie die Spieler getäuscht werden können. Mit diesem Projekt wollen wir diese Sicherheitslücke ein für alle Mal aus unserem Spiel verbannen

Planung

Zuerst haben wir uns einen groben Überblick über die Funktionsweise von AES verschafft. Dabei hat uns eine Aufnahme von einer Präsentation von Christof Paar sehr geholfen. Während der Umsetzung untersuchten wir dann jeder einzelne Schritt und setzten diesen dann um, sobald wir die Mechanik verstanden hatten.

Als wir dann mit dem Algorithmus fertig waren, implementierten wir diesen in unser Server/Client Modell des Spiels.

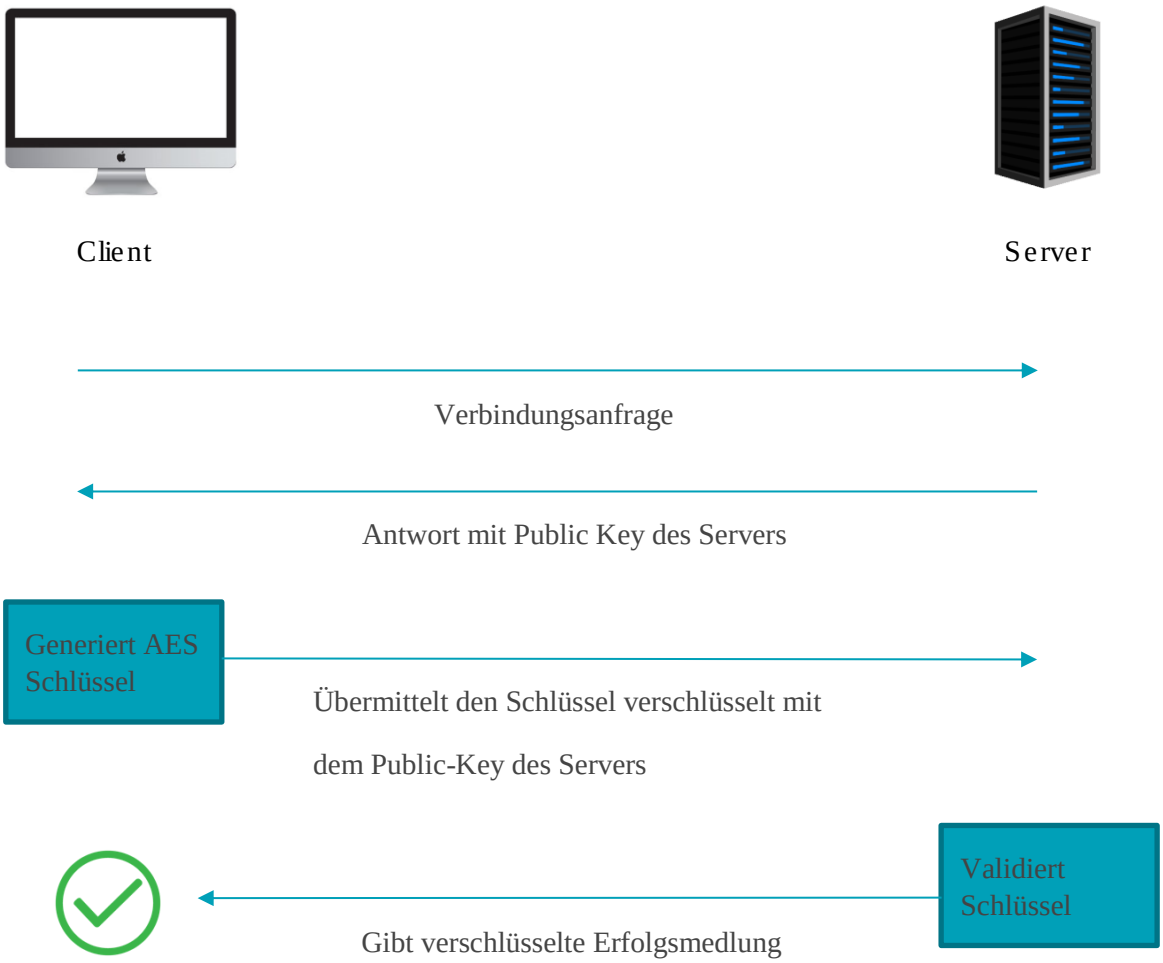
Verwendete Lösung

EINLEITUNG

Da eine fertige AES-Library zu verwenden langweilig wäre und keine Herausforderung bieten würden, haben wir uns dazu entschieden AES selber zu umzusetzen. Um zudem vollumfänglich eine sichere Kommunikation zu ermöglichen haben wir unser Krypto System mit den eingebauten Java-Libraries mit einem RSA Schlüsselaustausch erweitert.

PROGRAMMABLAUF

Verbindungsaufbau



Weiterer Ablauf

Nachdem erfolgreich die Verbindung aufgestellt wurde, schickt der Server dem Client alle verfügbare Spiele. Der Benutzer kann dann ein Spiel auswählen und diesem beitreten, wenn es nicht schon von zwei Spielern besetzt ist. Danach erscheint ein neues Fenster mit der GUI des Spieles: Ein einfaches Canvas, dass die Steine renderet. Durch klicken auf die einzelnen Spalten kann man einen Stein setzen, wenn man dran ist.

Sobald ein Spieler gewonnen hat, erkennt dies der Server und schickt eine Nachricht an beide Spieler, ob sie verloren oder gewonnen haben.

Networking

Die Kommunikation zwischen Server und Client läuft über ein TCP IPv4 Socket. Auf der Ebene der Anwendungsschicht haben wir ein eigens für dieses Projekt entwickeltes Protokoll, welches dann über den verschlüsselten Kanal übertragen wird.

Das Protokoll ist ein Request/Response Protokoll, es sieht also immer eine Antwort seitens des Servers auf eine Anfrage des Clients vor. Ein Request hat dabei die folgende Struktur:

Domain:Command:Parameter 1:Patameter 2…

Domain

> Der Domain kategorisiert den Request. Die verwendeten Kategorien sind bspw. Game, chat oder connection

Command

> Der Command spezifiziert, was der Request genau für eine Aktion innerhalb der Domäne ausführen soll. Verwendete Commands sind bspw. Connect, setStone oder joinGame

Parameter

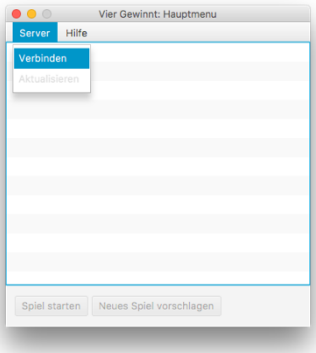
> In den Parametern lassen sich zusätzlich beliebig viele Informationen übermitteln, die für das Ausführen des Request notwendig sind. Ein Parameter ist bspw. Der Public-Key des Servers bei dem Verbindungsaufbau oder der Name des Clients beim Beitreten eines Spiels

Eine typische Antwort des Servers ist gleich aufgebaut wie der Request. Ein Beispiel wäre *connection:connect:success*, was die Nachricht ist, welche der Server verschickt, sobald die Verbindung mit dem Client erfolgreich hergestellt wurde.

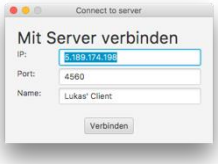
GRAPHICAL USER INTERFACE

Der Client besitzt eine GUI, welche mit FXML erstellt wurde. Die UX lässt sich in folgende Hauptteile unterteilen:

Das Hauptmenu

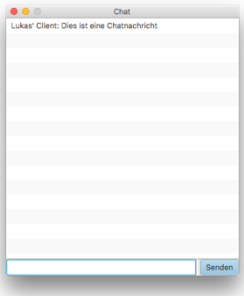


In dem Hauptmenu werden alle verfügbaren Spiele aufgelistet sowie über eine Toolbar die Funktion zum Verbinden mit dem Server angezeigt.



Das Verbinden wird ebenfalls über ein zusätzliches Fenster gehandhabt. Der Benutzer kann dort alle wichtigen Informationen für den Server angeben sowie einen Benutzernamen wählen

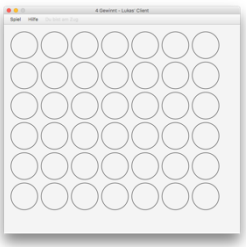
Der Chat



Das Programm bietet ebenfalls eine Möglichkeit zur Kommunikation zwischen den einzelnen Spielern in Form eines Chats an. Dabei wird die momentane Position des Benutzers im Spiel berücksichtigt: Wenn man sich im Hauptmenu befindet, kann man mit allen verfügbaren Spielern schreiben. In einem Spiel ist die Kommunikation zu dem Gegner limitiert.

Das Spiel

Das Spiel besteht aus einem grossen Fenster mit dem Spielbrett. Der Benutzer kann auf die jeweilige Spalte klicken, um einen Stein zu spielen. Über die Toolbar kann man weitere Einstellungen vornehmen, wie die Farbe der Steine zu setzen.



FUNKTIONSWEISE AES

AES besteht eigentlich nur aus fünf Methoden; KeyExpansion, ByteSubsitution, ShiftRow, MixColumn und AddRoundKey. Diese werden in unserem Fall bis zu 10 Mal auf den zu verschlüsselnden Text ausgeübt. Die KeyExpansion ist dafür verantwortlich aus dem Schlüssel sogenannte Rundenschlüssel zu generieren. Diese werden in den 10 Runden die durchlaufen werden verwendet. Die ByteSubsitution sorgt dafür, dass sich Bytes verändern und die ShiftRow und MixColumn sorgen dafür, dass sich diese Veränderungen auch über den gesamten Text verteilen und nicht nur an einem Ort sind. Die AddRoundKey Methode fügt, wie es der Name schon sagt, den Rundenschlüssel bitweise zu dem zu verschlüsselnden Text hinzu. Die erste und letzte Runde sind etwas anders als die «mittleren». Dies sieht man auch in den folgenden Bildern.

Anhang: Verwendete Dokumente und Notizen für die Planung

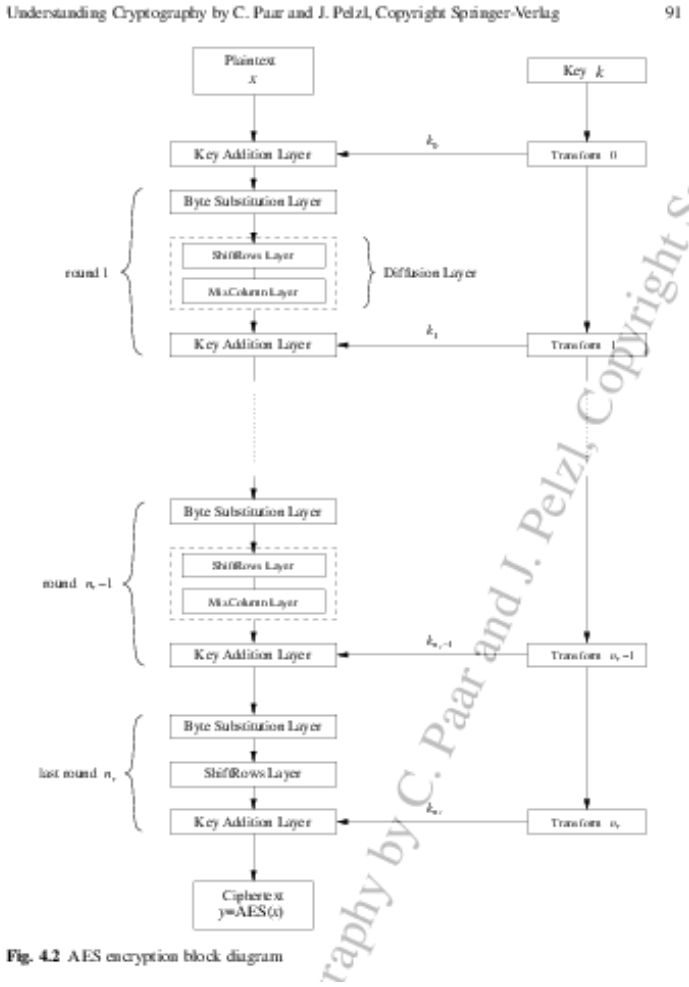
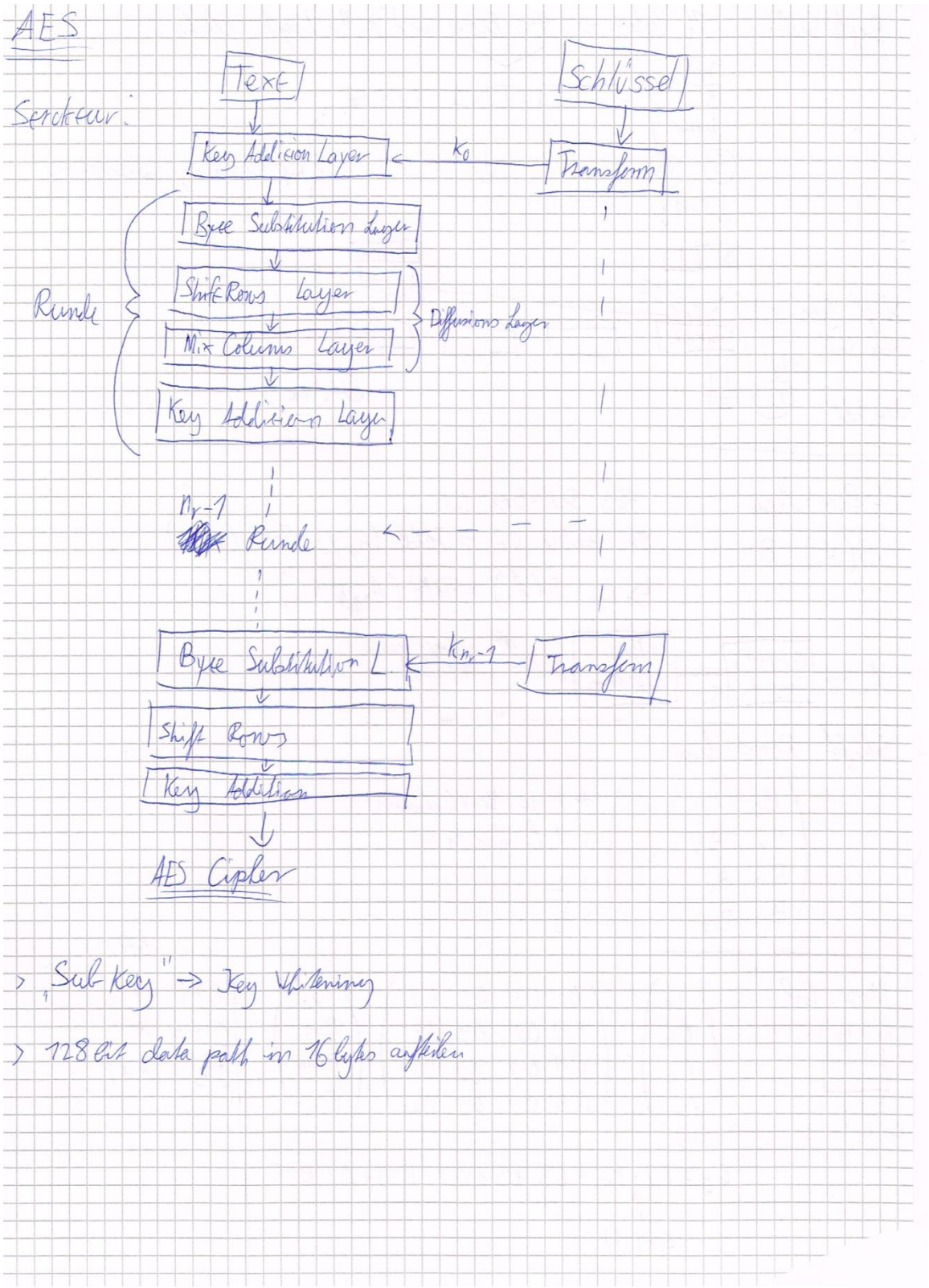


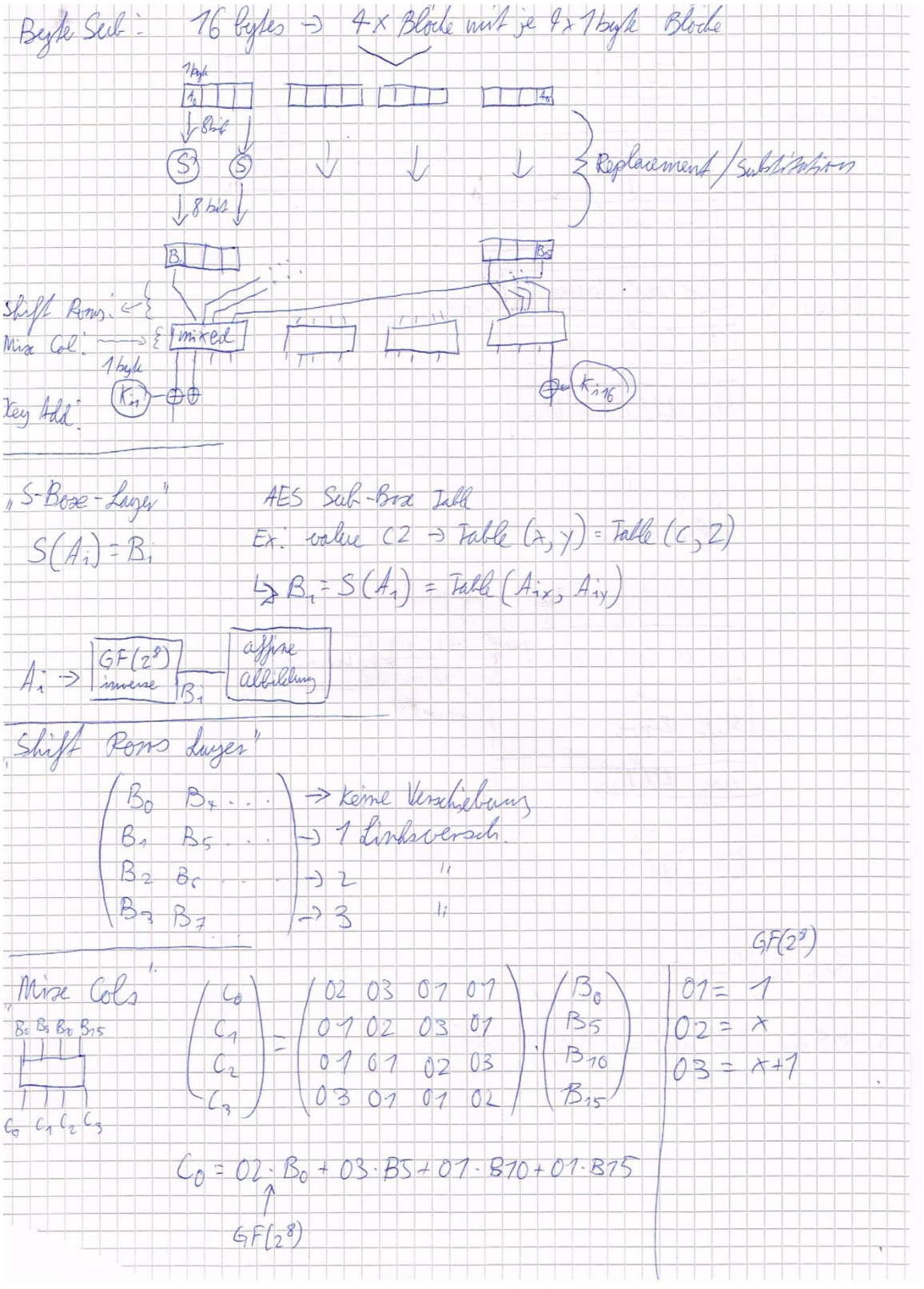
Fig. 4.2 AES encryption block diagram

Crypto-Textbook von Christoph Paar, crypto-textbook.com



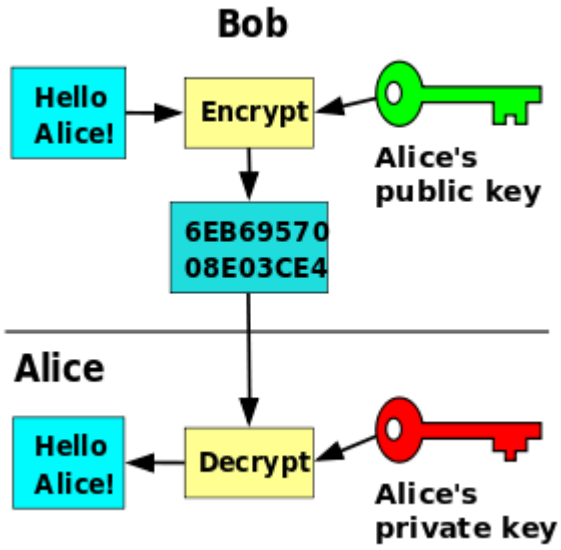
> „Sub-Key" → Key Whitening

> 128 bit data path in 16 bytes aufteilen


Byte Sub: 16 Bytes → 4x Blöcke mit je 4x1 byte Blöcke
1 byte
8 bits
Replacement / Substitution
Shift Rows:
Mix Col:
mixed
1 byte
Key Add:
„S-Box-Layer"
S(Ai) = Bi
AES Sub-Box Table
Ex: value C2 → Table (x, y) = Table (C, 2)
↳ Bi = S(Ai) = Table (Aix, Aiy)
Ai → GF(2^8) inverse → Bi → affine abbildung
„Shift Rows Layer"
B0 B4 ... → keine Verschiebung
B1 B5 ... → 1 Linksversch.
B2 B6 ... → 2 "
B3 B7 ... → 3 "
Mix Cols
B0 B5 B10 B15
C0 C1 C2 C3
GF(2^8)
01 = 1
02 = x
03 = x+1
C0 = 02·B0 + 03·B5 + 01·B10 + 01·B15
GF(2^8)

FUNKTIONSWEISE RSA

RSA funktioniert mit Public und Privat Keys. Der Public Key wird, wie es der Name bereits sagt, der Öffentlichkeit zugänglich gemacht. Mit diesem kann der andere Gesprächspartner, Bob, nun eine Nachricht verschlüsseln und an, in diesem Fall, Alice schicken. Alice kann die Nachricht, dann mit ihrem Private Key entschlüsseln.



In unserem Projekt generiert der Server einen Public- und einen Private-Key. Den Public-Key schickt er dem Client bei dem Verbindungsaufbau. Der Client generiert dann einen zufälligen AES Schlüssel und schickt diesen dann zum Server zurück, indem er die Nachricht mit dem Public-Key des Servers verschlüsselt. Dieser entschlüsselt die Nachricht dann mit seinem Private-Key und schickt dann eine Erfolgsmeldung AES-verschlüsselt mit dem erhaltenen Key zurück. Wenn der Client die Nachricht dann lesen kann, wird die Verbindung erhalten, ansonsten wird sie sofort getrennt.

Reflexion

Nachwort Philipp

Die Teamarbeit hat sehr gut funktioniert und wir haben uns die Arbeit sinnvoll und fair aufgeteilt. Leider haben sich ein paar unnötig dumme Fehler eingeschlichen, doch konnten wir diese, auch wenn mit erheblichem Zeitverlust, beheben.

Ich finde, dass es ein erfolgreiches Projekt ist, welches so funktioniert, wie wir uns das gewünscht haben.

Nachwort Lukas

In dem Projekt haben wir den Algorithmus sehr gut und koordiniert umgesetzt. Jedoch schlichen sich manchmal dumme Fehler ein, welche wir teilweise über eine Woche lang suchten und uns nach dem Finden dann dementsprechend über die Banalität aufregten.

Ich finde trotzdem, dass dies ein gelungenes Projekt ist, welches restlos nach unseren Wünschen funktioniert.