Projektaufgaben Kryptologie WS 14/15

Prof. Dr. rer. nat. habil. Alfons Geser
Fakultät EIT, HTWK Leipzig

Die Aufgaben für Programmierung auf Graphikprozessoren sollen in C oder C++ programmiert werden. Alle übrigen Programmieraufgaben sollen in Java oder Haskell programmiert werden. Öffentlich zugängliche Programmbibliotheken wie Java Cryptograhic Framework dürfen verwendet werden. Besonderer Wert wird auf einen sauberen Programmierstil gelegt.

Aufgabe 1: Knacken einer Digramm-Verschlüsselung

Ein Klartext, bestehend aus Kleinbuchstaben ohne Umlaute oder scharfem s, soll mit einer Digramm-Verschlüsselung verschlüsselt werden. Der Klartext soll dazu in 10 Fünfergruppen pro Zeile als Datei vorliegen. Der Schlüssel soll als 26-zeilige Datei mit 26 Buchstabenpaaren pro Zeile gegeben sein. Buhstabenpaare seien durch ein Leerzeichen voneinander getrennt. Alle fünf Buchstabenpaare soll ein zusätzliches Leerzeichen stehen, und alle fünf Zeilen soll eine Leerzeile kommen, damit ein Leser sich zurechtfinden kann. Mit einer Häufigkeitsanalyse von Buchstabenpaaren im Deutschen sollen Hinweise auf den Klartext gegeben werden.

Aufgabe 2: Knacken einer Transposition

Zu einem Geheimtext, der mit einer kolumnaren Transposition erzeugt worden ist, soll mit Hilfe eines bekannten Worts die Blocklänge bestimmt werden. Der Geheimtext soll dazu in 10 Fünfergruppen pro Zeile als Datei vorliegen. Anschließend sollen Stellen bestimmt werden, an denen das bekannte Wort im Klartext stehen könnte, und Folgerungen für den Schlüssel abgeleitet werden.

Aufgabe 3: AES-128 mit SHA-256 Rundenschlüsseln

Eine Variante des AES-128 soll den r-ten Rundenschlüssel folgendermaßen bilden:

1. S_0 sei der gegebene Schlüssel S

2. S_{r+1} entstehe durch Addition modulo 2^{32} der beiden Hälften des SHA-256-Streuwerts von S_r.

Programmieren Sie Verschlüsselung und Entschlüsselung mit dieser Variante durch Editieren einer Kopie der AES-128-Software und führen Sie an Beispielen vor, wie Verschlüsselung und Entschlüsselung von Blöcken arbeitet.

Aufgabe 4: AES-128 mit ARC4 Rundenschlüsseln

Eine Variante des AES-128 soll den r-ten Rundenschlüssel folgendermaßen bilden:

1. Alleged RC4 (ARC4) wird mit dem gegebeben Schlüssel S initialisiert.
2. S_r wird spaltenweise aufgefüllt mit den $(r+1)$-ten 16 Bytes, die ARC4 ausgibt.

Programmieren Sie Verschlüsselung und Entschlüsselung mit dieser Variante durch Editieren einer Kopie der AES-128-Software und führen Sie an Beispielen vor, wie Verschlüsselung und Entschlüsselung von Blöcken arbeitet.

Aufgabe 5: Quadratwurzel-Angriff

Auf die kommunizierten Daten eines Diffie/Hellman-Schlüsselaustauschs mit $n \leq 2^{32}$ soll mit dem Quadratwurzel-Angriff der gemeinsame Schlüssel ermittelt werden.

Mögliche Variante: Quadratwurzel-Angriff mit CUDA. $n \leq 2^{64}$. Implementierung auf einem Graphikprozessor (CUDA-Framework).

Aufgabe 6: Schlüsselstromgenerator mit SHA-256

Mit SHA-256 soll folgendermaßen ein Schlüsselstromgenerator implementiert werden. Der Zustand sei ein Bitvektor aus 256 Bits.

- Der Startzustand entsteht durch Anwenden von SHA-256 auf den Schlüssel.
- Der Folgezustand wird erzeugt durch Anwenden von SHA-256 auf das bitweise Komplement des derzeitigen Zustands.
- Ausgegeben werden die geradzahligen Bits des Zustands, die niederwertigen Bits davon zuerst. Rechtes Bit = Bit mit Position 0.

An Beispielen soll Verschlüsselung und Entschlüsselung mit der Stromchiffre vorgeführt werden. Weiter soll die Häufigkeit von Bitblöcken der Länge 4 bis 8 im Schlüsselstrom bestimmt werden.

Aufgabe 7: Lineare Kryptoanalyse

Implementieren Sie die Verschlüsselung in vier Runden, die in der Vorlesung als Beispiel zur Linearen Kryptoanalyse vorgestellt wurde. Die Rundenschlüssel seien in Form einer Tabelle fest vorgegeben. Erzeugen Sie 2^{16} gut gestreute Paare aus 16-Bit-Blöcken von Klartext und zugehörigem Geheimtext und führen Sie vor, wie Charlie daraus den Teilschlüssel ermittelt.

Aufgabe 8: Streufunktion mit AES-128

Mit AES-128 soll folgendermaßen eine Streufunktion mit Streuwertlänge 128 Bit implementiert werden:

1. Der aus einer Datei eingelesene Klartext wird wie in SHA-256 aufgefüllt.
2. Der aufgefüllte Klartext wird in Blöcke zu je 128 Bit zerlegt.
3. Das 128-Bit Textregister wird initialisiert wie die ersten 128 Bit in SHA-256.
4. Für jeden Klartextblock B wiederhole:
5. Der Inhalt des Textregisters wird mit AES-128 verschlüsselt, wobei B als Schlüssel dient.
6. Das Ergebnis der Verschlüsselung wird der neue Inhalt des Textregisters.
7. Ergebnis: Der Inhalt des Textregisters.

Die Streuwerte von einigen Beispielen sollen damit bestimmt werden.

Aufgabe 9: Streufunktion mit ARC4

Mit Alleged RC4 (ARC4) soll folgendermaßen eine Streufunktion implementiert werden:

1. Der aus einer Datei eingelesene Klartext wird wie in SHA-256 aufgefüllt.
2. Der aufgefüllte Klartext wird in Blöcke zu je 128 Bit zerlegt.
3. Das 256-Bit Textregister wird initialisiert wie in SHA-256.
4. Für jeden Klartextblock B wiederhole:
5. Die niederwertigen 128 Bits des Textregisters werden bitweise EXOR-verknüpft mit B. Das liefert die neuen 128 Bits des Textregisters.
6. Mit dem Textregister wird ARC4 initialisiert.
7. Man läßt die nächsten 256 Bytes des ARC4 verfallen.
8. Die nächsten ausgegebenen 256 Bit des ARC4 liefern den neuen Inhalt des Textregisters.
9. Ergebnis: Der Inhalt des Textregisters.

Die Streuwerte von einigen Beispielen sollen damit bestimmt werden.

Aufgabe 10: Geburtstagsangriff

Der Geburtstagsangriff auf Streufunktionen soll implementiert werden. Eine Streufunktion mit 32 Bit Streuwertlänge soll gebildet werden durch bitweises EXOR der acht 32-Bit-Blöcke des Ergebnisses der SHA-256 Streufunktion. Diese Streufunktion soll mit einem Geburtstagsangriff folgendermaßen geknackt werden. Schreiben Sie zwei gegensätzliche Rezensionen zu einen frei gewählten künstlerischen Ereignis. Beide Rezensionen sollen 16 Schablonen enthalten, die mit je 2 möglichen Worten belegt werden können. Bestimmen Sie zu jeder Belegung die Streuwerte und testen Sie auf Kollisionen.

Mögliche Variante: Geburtstagsangriff mit CUDA. 64 Bit Streuwertlänge. Implementierung auf einem Graphikprozessor (CUDA-Framework).

Aufgabe 11: Vortrag: Kasiski-Test

Die Funktionsweise und der mathematische Hintergrund des Kasiski-Tests sollen in einem halbstündigen Vortrag erklärt werden.

Aufgabe 12: Vortrag: Serpent-Kryptosystem

Die Funktionsweise und die Sicherheitsaspekte des Serpent-Kryptosystems sollen in einem halbstündigen Vortrag erklärt werden.

Aufgabe 13: Vortrag: ARC4

Die Funktionsweise und die Sicherheitsaspekte des ARC4 (Alleged RC4) Schlüsselstromgenerators sollen in einem halbstündigen Vortrag erklärt werden.

Aufgabe 14: Vortrag: Erzeugung der Rundenschlüssel in AES-128

In einem halbstündigen Vortrag soll der Algorithmus zur Erzegung der Rundenschlüssel in AES-128 aus der Vorlesung eingehend mit der Beschreibung in der offiziellen AES-Spezifikation verglichen werden.

Aufgabe 15: Vortrag: Quanten-Kryptographie

Die Funktionsweise und die Sicherheitsaspekte des Schlüsselaustauschs mittels Lichtquanten sollen in einem halbstündigen Vortrag erklärt werden.

Aufgabe 16: Vortrag: DSS-Unterschriften

Die Funktionsweise und die Sicherheitsaspekte des Digital Signature Standards (DSS) sollen in einem halbstündigen Vortrag erklärt werden.

Aufgabe 17: Vortrag: Quadratwurzel-Angriff

Der Quadratwurzel-Angriff gegen den Diffie/Hellman-Schlüsselaustausch soll in einem halbstündigen Vortrag erklärt werden.