# Inhaltsverzeichnis



# 1 Einführung und Grundlagen

## 1.1 Aufgabenstellung

## 1.2 Grundlagen

### 1.2.1 AES im Kurzüberblick

Da der Data Encryption Standard aufgrund seiner geringen Schlüsselgröße und der stark angestiegenen Rechenleistungen als zu schwach eingestuft worden war, musste ein neuer

Verschlüsselungsstandard her. Diesbezüglich hat die US Regierung im Jahre 1997 ein Wettbewerb abgehalten zu dem sich viele Arbeitsgruppen angemeldet hatten und Ihre Lösungen präsentierten.

Die Algorithmen wurden auf folgende Kriterien hin untersucht:

- Sicherheit
- Implementierbarkeit
- Ressourcenverbrauch / Speicherbedarf
- Geschwindigkeit

Als Gewinner wurde eine leicht veränderte Version des Rijndael, nach den Entwicklern Joan Daemen und Vincent Rijmen benannt, ausgewählt.

Der Algorithmus sei mit den heute zur Verfügung stehenden Mitteln nicht zu knacken und erlaube dank seines einfachen Aufbaus eine sehr schnelle Implementierung in unterschiedlichen Programmiersprachen. Dank der Einfachheit sei es auch sehr effizient und biete eine gute bis sehr gute Performance auf verschiedenen Plattformen, wie einem 32Bit Prozessor, 8Bit Mikrocontroller und auch eine Implementierung in Hardware. Weiterhin benötigt der Algorithmus sehr wenig Ressourcen und ist damit ideal für die Anwendung in Umgebungen mit beschränkten Ressourcen, wie z.B. Chipkarten.

Der Rijndael ist ein Blockchiffre, d.h. der Algorithmus arbeitet mit einer fest vorgegebenen Bitgruppe arbeitet, einem sogenannten Block. Die gewöhnliche Größe eines Eingabeblocks beträgt 128Bit welcher in einen dazugehörigen Ausgabeblock gleicher Größe verschlüsselt wird. Dazu wird ein Schlüssel gebraucht. Bei Rijndael ist es theoretisch möglich einen Schlüssel beliebiger Größe zu nehmen, der AES jedoch nimmt nur die Schlüsselgrößen 128Bit, 192Bit und 256Bit an.

Rijndael verwendet zur Verschlüsselung eine umkehrbare 8x8Bit Matrix, genannt Substitution Box (kurz S-Box) und Berechnungen über einem Galoisekörper der Form GF[28]. Diese Berechnungen erinnern an Fehlererkennende und –korrigierende Codes.

Um einen Block zu verschlüsseln, werden verschiedene Operationen nacheinander auf den Block angewendet, nämlich Substitutionen (S-Box) und Permutationen (P-Box). AES beruht damit auf ein Substitutions- und Permutations- Netzwerk (SPN). Der AES führt die genannten Operationen iterativ auf je einen Block aus. Ein Block muss sequenziell verarbeitet werden, aber es können mehrere Blöcke gleichzeitig verarbeitet werden, da diese unabhängig voneinander sind. Dieser Effekt ist ideal für die Nutzung des Algorithmus auf einer GPU, da diese sehr viele Operationen Parallel verarbeiten kann. Es können damit viele Blöcke Parallel verschlüsselt werden und dies führt zu einem theoretisch enormen Speedup.

Wie schon oben erwähnt ist eine Rijndael ein Blockchiffre, wobei die Blocklänge und die Schlüssellänge unabhängig voneinander variieren können. Es können die Werte 128 (min), 160, 192, 224 oder 256 Bits gewählt werden. Bei AES jedoch sind die Blockgrößen auf 128 Bit beschränkt, die Schlüsselgröße kann aber die Werte 128, 192 oder 256 Bit annehmen. Jeder Block wird im Rijndael in eine zweidimensionale Tabelle mit vier Zeilen

geschrieben, wobei jede Zelle die Größe ein Byte hat. Die Anzahl der Spalten ist bei AES konstant und beträgt 4 (128Bit).

### 1.2.2 CUDA Framework

Standard Das „Compute Unified Device Architecture Software Developer Kit“ (CUDA SDK) wurde von NVIDA am 15. Februar 2007 das erste mal der Öffentlichkeit vorgestellt. Ziel dieses SDKs ist es, eine parallele Ausführung von Code auf unterstützten Grafikkarten. Zur Zeit sind das die aktuellen Grafikkarten, welche mit einem GeForce, ION, Quadro oder Tesla Grafikprozessor ausgestattet sind.

Standard CUDA basiert auf eine abgewandelten Variante von C. Typischerweise wird bei CUDA Anwendungen die Busbandweite und Latenz zwischen CPU und GPU zum Engpass. Darüber hinaus erreicht man die optimale Geschwindigkeit nur, wenn man die Implementierung an die Hardware anpasst (z.B. sollte die Anzahl der parallellaufenden Threads gleich die Anzahl der Streaming -Prozessoren sein).

Standard CUDA ist weitestgehend plattformunabhängig. So ist es möglich, „CUDA-Programme“ auf Windows, Linux und Mac OSX auszuführen - eine kompatible Grafikkarte vorausgesetzt.

# 2 GPU Architektur

## 2.1 Historie

Ursprünglich war es die einzige Aufgabe einer Grafikkarte, ein Bild auf einem Anzeigegerät (wie z.B. einem Monitor) darzustellen. Im Laufe der Geschichte haben sie sich in programmierbare Prozessoren entwickelt. Aufgrund ihrer Geschichte aus der Grafikberechnung, sind Grafikarten in der Lage, sehr viele Operationen parallel zu berechnen. Wie genau das von statten geht, wir im Kapitel der Architektur erwähnt. Durch Ausnutzen der Parallelität der Grafikkarte in Berechnungen kann gegenüber der Berechnung mit der CPU schneller berechnet werden.

## 2.2 Architektur

### 2.2.1 Prozessoraufbau

Moderne Grafikarten sind über die PCI-E Schnittstelle an die CPU angebunden. Über diesen Bus werden die Daten und Prozesse an die berechnenden Einheiten der GPU übertragen.

Im Folgenden ist die Architektur der GPU abgebildet. Hierbei ist zu beachten, dass die Anzahl der Multiprozessoren auf der GPU, sowie die Anzahl der Streaming-Prozessoren (SP) je nach Modell unterschiedlich sind.

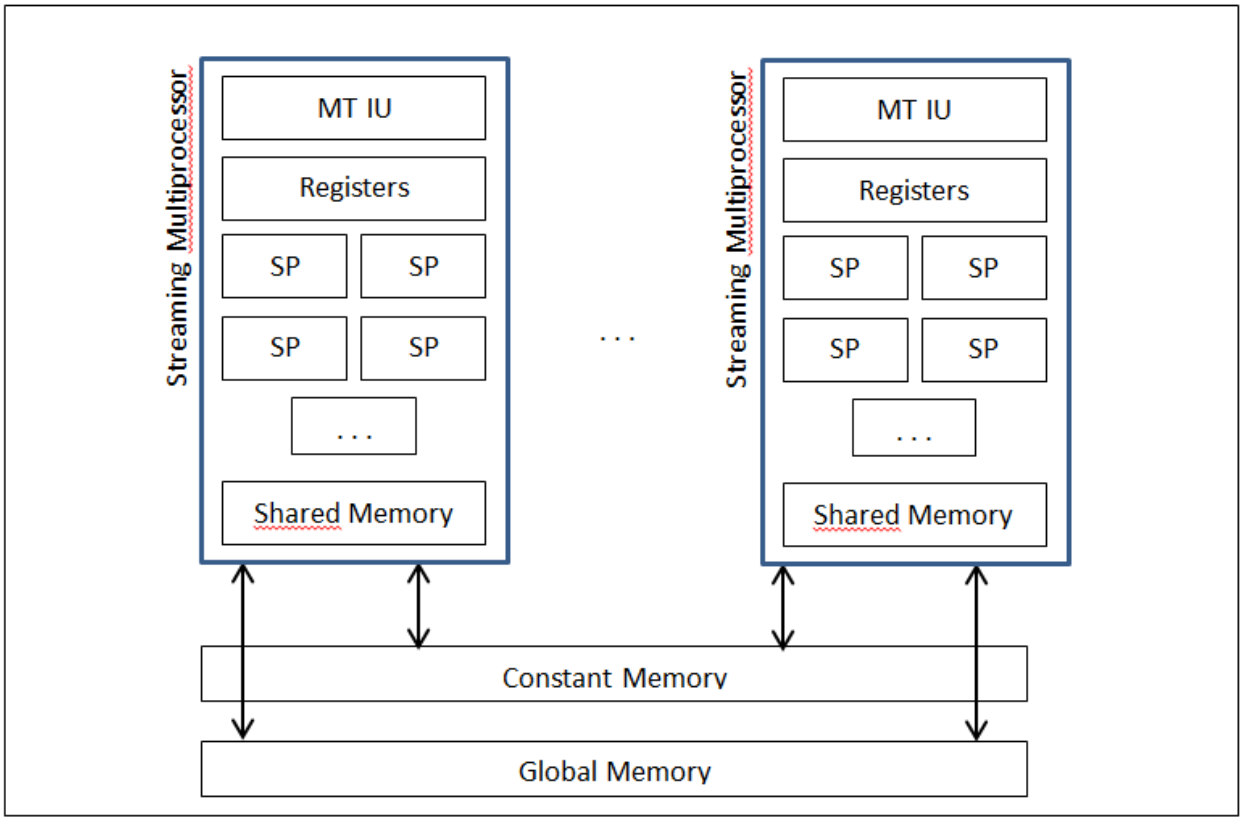


Abbildung 1: GPU-Architecture

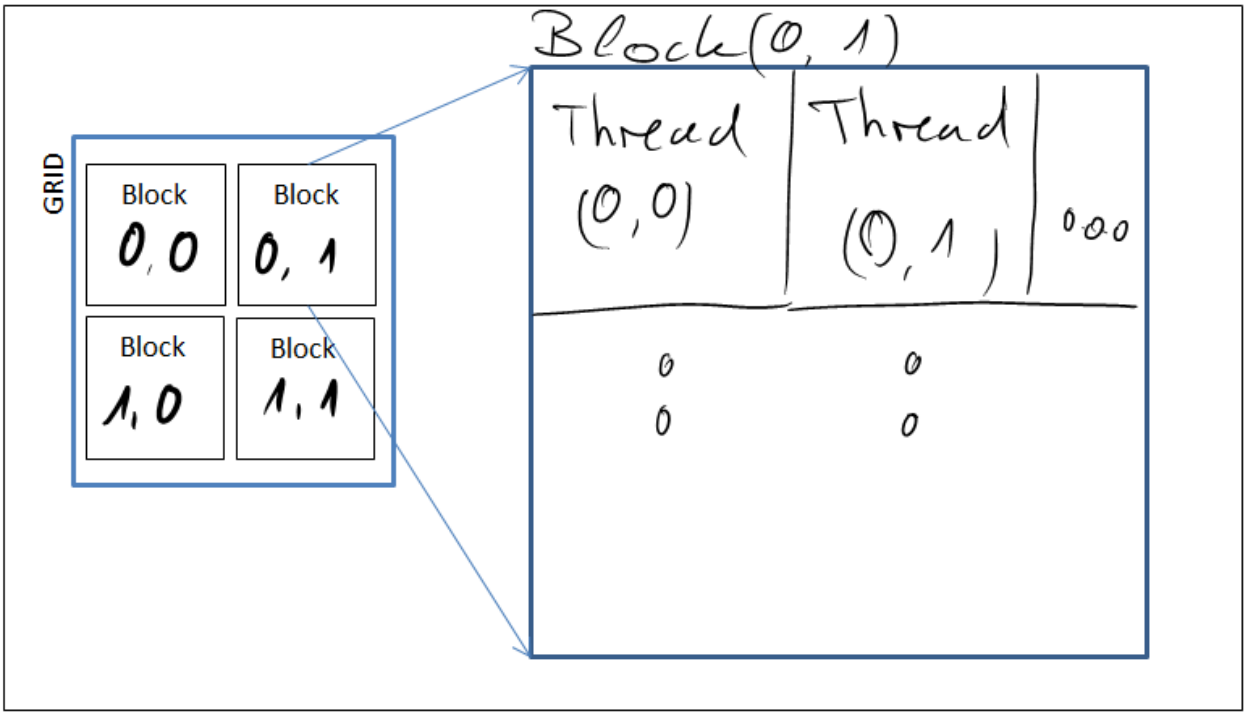


Abbildung 2: Streaming Processor

### 2.2.2 Multithreaded Instruction Unit (MT IU)

Die MT IU verwaltet die Ausführung von Threads auf dem Multiprocessor. Hierbei werden einzelne Threads zu einem Block, mehrere Blöcke zu einem Grid zusammengefasst.

### 2.2.3 Streaming Multiprocessor

Auf jeden Streaming Multiprocessor wird genau ein Block abgebildet. Die in dem Block befindlichen Threads werden -soweit möglich- parallel abgearbeitet. Hierbei hat jeder Thread in dem Block eine eindeutige ID, auf welche auch in dem Thread zugegriffen werden kann.

### 2.2.4 Streaming Processor (SP)

Jeder SP führt genau einen Tread aus.

## 2.3 Speicherhierarchie

### 2.3.1 Global Memory

Der Global Memory (RAM) ist der Größte Bereich und sowohl von CPU als auch von der GPU schreib- und lesbar. Dieser Speicher ermöglicht den Austausch von Daten zwischen

GPU und CPU. Dieser Speicher hat die größte Kapazität, ist jedoch der langsamste der Speicherhierarchie bezüglich der GPU.

#### 2.3.2 Constant Memory

Der Constant Memory ist physikalisch auf 64KB beschränkt.

#### 2.3.3 Register

Threads eines Bocks teilen sich gemeinsame Register.

#### Shared Memory

Für jeden Streaming-Multiprozessor-Prozessor ist ein Shared Memory vorgesehen, welches der schnellste -mit 16 KB jedoch auch der kleinste- Speicher in der Hierarchie der GPU ist. Der Multiprozessor teilt diesen verfügbaren Speicher und seinen Streamingprozessoren auf. Dieser Speicher kann nur von der GPU gelesen und geschrieben werden.

# 3 Implementierung

## 3.1 Funktionen im Detail

### 3.1.1 Galois-Feld-Theorie etc.

### 3.1.2 MixColumn-Funktion

### 3.1.3 Substitutionsbox

### 3.1.4 ShiftRow-Funktion

## 3.2 CUDA-spezifische Veränderungen

### 3.2.1 Prozessaufteilung

### 3.2.2 Speichernutzung

# 4 Tests und Benchmarks

## 4.1 Testumgebung

## 4.2 Ergebnisse

# 5 Ausblick