Effizienteres Bruteforcing auf einem heterogenen Cluster mit GPUs und FPGAs*

J. Fuß, B. Greslehner-Nimmervoll, W. Kastl, R. Kolmhofer

Department Sichere Informationssysteme, FH OÖ
Softwarepark 20
A-4232 Hagenberg im Mühlkreis
{juergen.fuss, robert.kolmhofer}@fh-hagenberg.at
{bernhard.greslehner-nimmervoll, wolfgang.kastl}@fh-hagenberg.at

Abstract: Heterogene Cluster mit verschiedenen Coprozessor-Typen können homogenen Lösungen überlegen sein. Diese Arbeit behandelt die Implementierung von verteilten, kryptoanalytischen Aufgaben in einer heterogenen Umgebung. Die Verwaltung der Komponenten übernimmt ein eigenständiges Cluster-Framework. Es wird verwendet, um passwortgeschützte PDF-Dokumente – verteilt auf CPUs, GPUs und FPGAs – zu brechen. Zusätzlich zeigen die Ergebnisse, dass ein Verwaltungs-System für heterogene Hardware keine beträchtlichen Leistungs-Einbußen verursachen muss.

1 Einleitung

Aufwändige kryptoanalytische Aufgaben werden üblicherweise parallelisiert gelöst – entweder in verteilten Systemen mit sehr wenig oder gar keinem Kommunikationsbedarf oder in Cluster-Knoten mit ähnlicher oder sogar gleicher Prozessorarchitektur. In den folgenden Abschnitten wird ein Ansatz vorgestellt, der verschiedene Coprozessoren verwendet, um die Vorteile eines heterogenen Systems effizient zu nutzen. Ein Vorteil eines solchen Systems besteht darin, dass unterschiedliche Aufgaben auf dem jeweils dafür optimalen System gelöst werden können. Komplexe Aufgaben können in Teilaufgaben unterteilt und verschiedenen Coprozessoren zugewiesen werden, um eine höhere Leistung zu erzielen. Dieser Ansatz kann aufgrund der Modularität zusätzlich Entwicklungszeit einsparen. In dieser Arbeit wird der Nutzen dieser Methode demonstriert, indem das Passwort eines geschützten PDF-Dokuments durch einen Brute-Force-Angriff ermittelt wird. Zuerst wird der benötigte Schlüssel auf Grafikkarten berechnet und später auf FPGAs parallel validiert.

*Diese Arbeit wurde finanziert im Sicherheitsforschungs-Förderprogramm KIRAS vom Bundesministerium für Verkehr, Innovation und Technologie. Wesentliche Teile dieser Arbeit wurden bereits in [DFG+13] veröffentlicht.

2 Themenbezogene Arbeiten

Es existieren mehrere Arbeiten, die die Benutzung von Grafikprozessoren (GPUs) und Field Programmable Gate Arrays (FPGAs) zur Beschleunigung von kryptographischen Anwendungen behandeln. In diesem Kontext von Bedeutung ist speziell [LWC+09], welches die Bruteforce-Attacke von PDF-Verschlüsselung auf GPUs beschreibt. Ergänzend existieren mehrere Ansätze [WYWS12, HMH09, WLT11], welche die Beschleunigung von MD5 auf GPUs behandeln, sowie die effiziente Implementierung von RC4 auf FPGAs [KL08]. Die Ergebnisse der genannten Arbeiten flossen in die Entwicklung des hier vorgestellten PDF-Passwort-Crackers ein. Im Projekt AXEL wird ein ähnlicher Ansatz verfolgt, um klassische High-Performance-Computing-Probleme zu lösen [TL10].

3 Entschlüsselung passwortgeschützter PDF-Dokumente

Diese Arbeit behandelt ein gängiges Verschlüsselungsverfahren des Portable-Document-Format(PDF)-Standards nach ISO 3200 [Ado08]. Es verwendet eine Kombination von MD5 und RC4 für die Verbesserung der Schlüsselsicherheit und RC4 oder AES für die eigentliche Verschlüsselung. Dies ist die Standardeinstellung für eine Vielzahl von PDF-Generatoren.

Abbildung 1 zeigt den kompletten Algorithmus in schematischer Form. Der erste Schritt besteht in der Berechnung des Encryption Keys. Dieser Schlüssel ergibt sich durch mehrfaches MD5-Hashen des Benutzerpassworts, welches mit verschiedenen Parametern aus dem PDF-Header – P-Entry, O-Entry, File-ID und ein konstanter Initialwert – kombiniert wird. Der Paramter P-Value spezifiziert, welche Zugriffrechte auf das PDF-Dokument gewährt werden. O-Entry ist ein 32-Byte-String und vom Benutzerpasswort abgeleitet. Dieser wird benutzt um die Korrektheit eines beim Öffnen des Dokuments eingebenen Passworts festzustellen. Der zeitaufwändigste Teil ist die Berechnung der 51 MD5-Runden, wovon jede den Ausgabewert der vorherigen MD5-Runde als Eingabeparameter benötigt. Der Encryption Key wird für die eigentliche Dokumentenverschlüsselung verwendet. Er ist die Basis für die Berechnung des User Keys. Der User Key wird für die Überprüfung eines eingegebenen Passworts eingesetzt. Eine Kopie dieses Schlüssels ist im PDF-Header für die Passwortvalidierung gespeichert. Wenn ein Benutzer ein Passwort zur Dokumentenentschlüsselung eingibt, wird der User Key berechnet und mit dem im Dokument gespeicherten verglichen. Sind beide identisch, wurde das richtige Passwort eingeben. Die implementierte Bruteforce-Attacke verwendet diesen Mechanismus, um das korrekte Passwort zu ermitteln. Der User Key wird berechnet, indem ein MD5-Hash der File-ID erzeugt und 20-mal mit RC4 verschlüsselt wird. Der Encryption Key dient als Eingabe. Der Hash der File-ID kann im Voraus berechnet werden, um Zeit zu sparen. In Summe benötigt eine einzelne Passwortvalidierung 51 MD5-Berechnungen und 20 RC4-Verschlüsselungen.

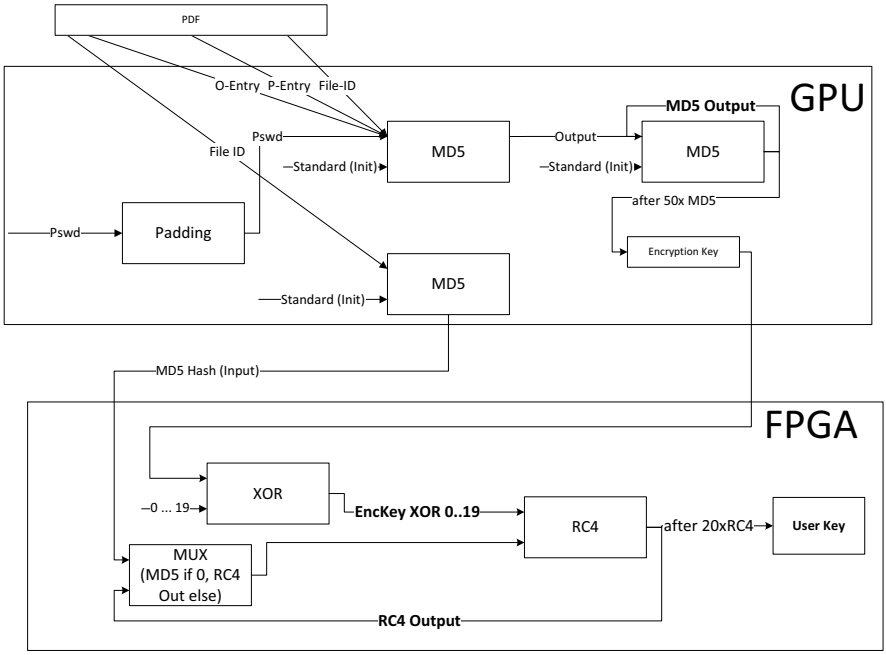

Abbildung 1: Ablauf der PDF Entschlüsselung.

4 Implementierung

Die Berechnung eines User Keys wird auf zwei verschiedenen Coprozessor-Typen ausgeführt. Die GPUs generieren die Encryption Keys mit Hilfe des MD5-Algorithmus. Die Ausgabe wird an die FPGAs weitergeleitet. Dort werden die mit RC4 verschlüsselten User Keys mit dem Schlüssel des PDF-Headers verglichen (siehe Abbildung 1). Diese Aufteilung wurde aufgrund der Eignung der jeweiligen Coprozessoren durchgeführt. Durch die wenigen Speicherzugriffe (außerhalb von Registern) ist der MD5-Algorithmus für eine Berechnung durch GPUs geeignet. Der RC4-Algorithmus hingegen ist nur mit Leistungs-Einbußen auf GPUs ausführbar. Die großen, intern im Algorithmus verwendeten 256-Byte-Arrays passen nicht in die relativ kleinen, schnellen Speichereinheiten der GPU. Daher müsste auf den größeren aber um ein Vielfaches langsameren, globalen Speicher ausgewichen werden. Dies wurde mit Messungen bestätigt: Auf einer NVIDIA-Geforce-GTX-680-Grafikkarte konnte ein Durchsatz von 17,16 Millionen Encryption Keys pro Sekunde erzielt werden. Dagegen wurden lediglich 950.000 RC4-verschlüsselte User Keys pro Sekunde erreicht.
Insgesamt erreicht die komplette GPU-Implementierung des Algorithmus einen Durchsatz von 680 kKps (Tsd. Schüssel pro Sekunde) auf einer NVIDIA-Geforce-GTX-680-Grafikkarte.

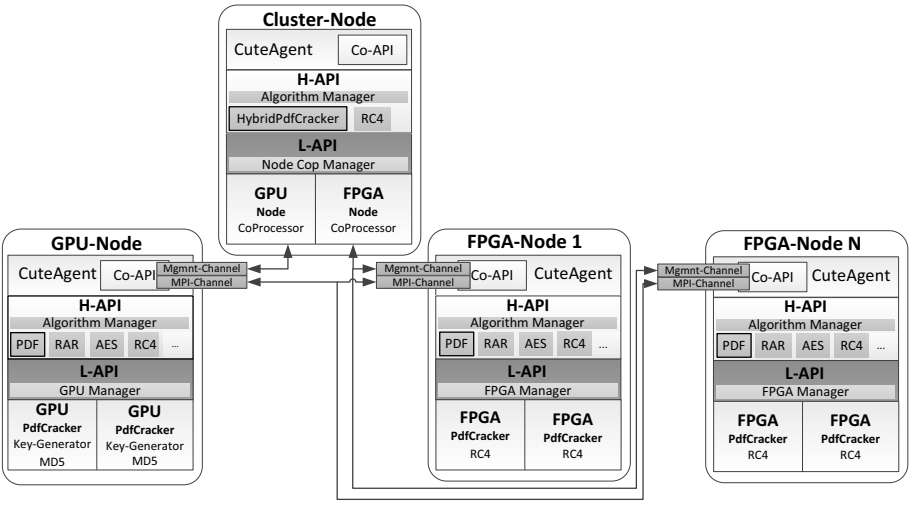

Abbildung 2: Schematischer Auftbau der Hard- und Software-Komponenten inklusive Datenfluss.

4.1 Cluster-Verwaltung

Das System verwendet ein eigenständiges Cluster-Framework um die Verteilung der Teilaufgaben und den Datenfluss durch den Cluster zu steuern (siehe Abbildung 2). Es besteht aus zwei Basiskomponenten: dem CuteMaster, welcher als Kontrollinstanz auf der Management-Node des Clusters ausgeführt wird und dem CuteAgent, welcher als Message Passing Interface (MPI)-Job auf den Compute-Nodes verteilt wird. Der CuteMaster konfiguriert das System mittels Konfigurationsdatei oder Benutzereingabe und überträgt die notwendigen Informationen durch den Management-Channel (via Transmission Control Protocol (TCP) über Ethernet) an die CuteAgents.

Sobald die Parameter – Pfad zur PDF-Datei; Passwortmaske; Anzahl der Blöcke, auf die der Passwortsuchraum aufgeteilt wird; Adressen der Clusternodes – übermittelt wurden, verbinden sich die CuteAgents mit den vorher zugewiesenen Coprozessoren, initialisieren sie und öffnen die MPI-Verbindungen für die Kommunikation über Infiniband mit den anderen Cluster-Nodes. In der ersten Phase werden Encryption Keys auf den GPU-Nodes erzeugt. Mehrere Schlüssel werden zu Schlüsselblöcken zusammengefasst und gleichmäßig auf die FPGA-Nodes verteilt. Dort wird die Berechnung mit der Generierung der User Keys fortgesetzt, welche abschließend mit dem User Key des PDF-Dokuments verglichen werden.

4.2 GPU-Teil

Eine NVIDIA GeForce GTX 680 GPU berechnet die MD5-Hashes. Diese wurde wegen ihrer Verfügbarkeit, niedrigen Hardwarekosten und fortgeschrittenen Entwicklungswerkzeuge ausgewählt. Als Programmiersprache kommt C for CUDA zum Einsatz. Der NVIDIA-Grafikkartentreiber führt in Zusammenarbeit mit der CUDA Runtime Library Speichertransfers mit einer Übertragungsgeschwindigkeit von ca. 4,6 GBps zu und von der GPU durch.

Um die Wiederverwertbarkeit des Programmcodes zu steigern, wurde die Berechnung auf den GPUs in drei Kernels aufgeteilt. Der erste Kernel generiert die Passwörter in Übereinstimmung mit einer festgelegten Passwortmaske. Der zweite übernimmt das Padding. Der dritte Kernel hasht die Passwörter samt Padding mit dem MD5-Algorithmus. So kann z. B. die Passwortgeneratorkomponente ausgetauscht werden.

4.2.1 Passwort-Generierung

Für die Passwortgenerierung wird ein einfacher Generator verwendet, welcher Passwörter auf Basis einer Passwortmaske erstellt. Diese Maske erlaubt einzelne Zeichentypen für jede Passwortposition. Z.B. ist es möglich, Passwörter zu generieren, welche mit zwei Zahlen beginnen und von vier Buchstaben und zwei Sonderzeichen gefolgt werden. Diese feine Kontrolle ist nützlich, wenn Informationen über das Passwort vorhanden sind.

Die erzeugten Passwörter werden im globalen Speicher auf der GPU so gespeichert, dass optimale Zugriffsmuster (coalesced access) angewendet werden können.

4.2.2 Padding

Die generierten Passwörter werden mit einem Padding String – dem PDF-Standard [Ado08] entsprechend – gefüllt, sodass sie exakt 32 Bytes lang sind.

4.2.3 Hashing

Aufgrund der seriellen Architektur des MD5-Algorithmus berechnet jeder GPU-Thread alleine einen kompletten MD5-Hash. Der GPU-Thread lädt das gepaddete Passwort vom globalen Speicher und hasht dieses mit dem O-Wert (Owner Key), welcher im konstanten Speicher abgelegt ist.

Der MD5-Algorithmus ist als optimiertes Macro implementiert, welches garantiert, dass oft benutzte Variablen in Registern abgelegt werden. Sogar mit dieser Maßnahme bleibt der Ressourcenverbrauch des Kernels niedrig. Wird für die neueste GPU-Generation (Compute Capability 3.0) kompiliert, so werden 30 Register und 87 Bytes *constant memory* benötigt. Der niedrige Ressourcenverbrauch ergibt ein *occupancy* von 100 %. Das bedeutet, dass der GPU-Scheduler so viele Threads wie möglich verwaltet, um Speicherzugriffsverzögerungen durch intelligentes Context-Switching zu verbergen. Der GPU-Thread holt

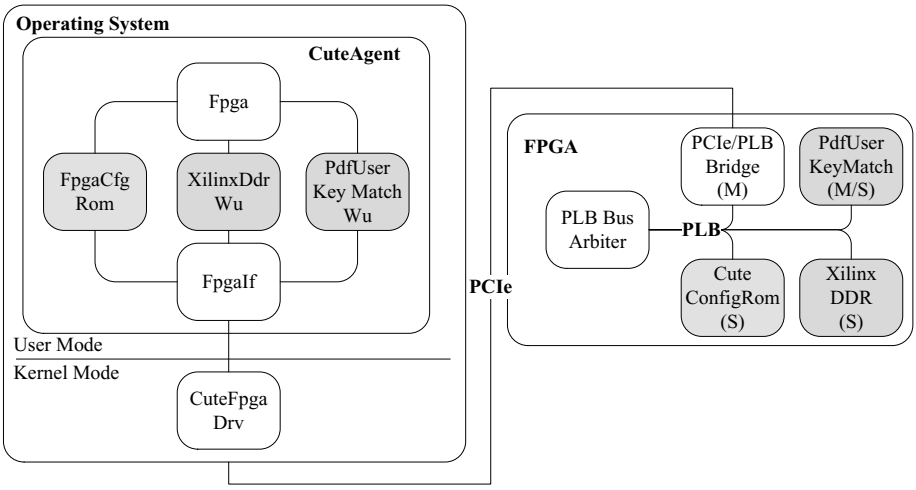

Abbildung 3: Übersicht FPGA-Node.

sich nur zu Beginn Daten aus dem langsamen, globalen Speicher. Daher kann angenommen werden, dass die GPU maximal ausgelastet ist und keine Leerlaufzeiten aufgrund von kostspieligen Speicheroperationen entstehen.

4.3 FPGA-Teil

Als Plattform wurde das Xilinx-ML605-FPGA-Entwicklungs-Board (ML605) verwendet. Die entwickelte Intellectual Property (IP) wurde im Xilinx Embedded Development Kit (EDK) als Buskomponente hinzugefügt. Aufgrund der Einfachheit und Stabilität wurde zur Kommunikation der Peripheral Local Bus (PLB) gewählt, welcher ursprünglich von IBM spezifiziert wurde. Auf den Einsatz des Advanced eXtensible Interface Bus (AXI) wurde verzichtet, da zur Zeit der Entwicklung die Unterstützung in der Xilinx Tool Chain nicht ausreichend war. Die Verbindung zwischen Host und ML605 wurde mit der Peripheral Component Interconnect Express (PCIe)-Schnittstelle realisiert.

In Abbildung 3 sind die am FPGA implementierten Teile der PDF-Entschlüsselung dargestellt. Das System besteht aus zwei Hauptblöcken, der Kontrollsoftware im CuteAgent und dem FPGA System on a Chip (SOC). Die beiden Komponenten sind via PCIe und PCIe-Treiber verbunden.

4.3.1 Infrastruktur

Die Infrastruktur wird hauptsächlich vom Softwaremodul „Fpga“ verwaltet. Dieses bewerkstelligt die Enumeration der FPGA-Boards. Die FPGA-Konfiguration wird herausge-

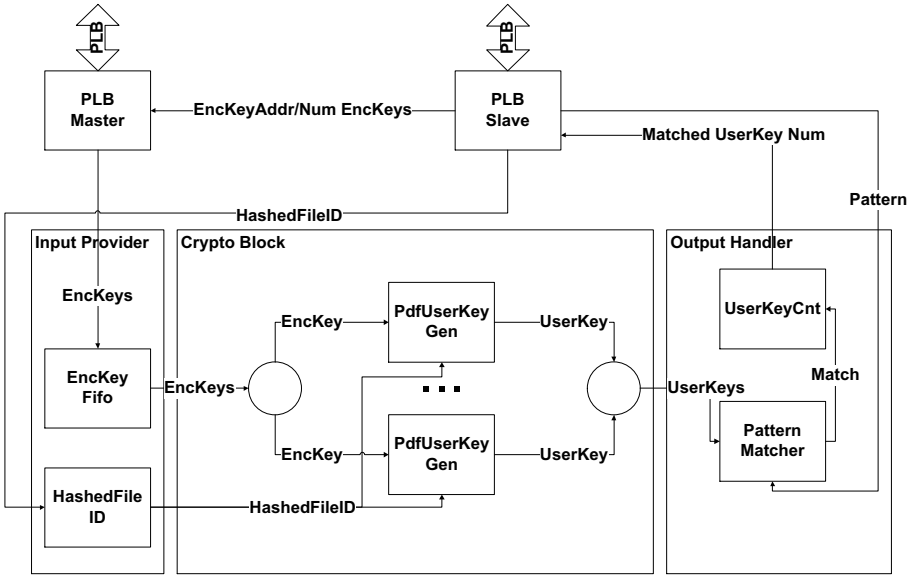

Abbildung 4: Struktur des „PdfUserKeyMatch“-IP.

funden, indem das SW-Modul „FpgaCfgRom“ die Informationen vom Hardware-IP „CuteConfigRom“ ausliest. Das Read Only Memory (ROM) enthält die Anzahl der im SOC befindlichen IPs ebenso wie deren Identifikation und SOC-Busadressen. Anschließend werden die dazugehörigen SW-Module geladen.

4.3.2 Entschlüsselungsprozess

Die FPGA-IPs für den eigentlichen Entschlüsselungsprozess sind der Xilinx Double Data Rate (DDR) Block und der „PdfUserKeyMatch“. Zuerst werden die von der GPU generierten Encryption Keys in das DDR RAM auf das ML605 transferiert. Sobald dieser Vorgang abgeschlossen ist, liest die „PdfUserKeyMatch“-IP die übertragenen Schüssel aus dem RAM und berechnet die entsprechenden User Keys. Falls der berechnete User Key mit dem in der PDF-Datei hinterlegten übereinstimmt, wird ein Flag gesetzt und der Index des gefundenen Encryption Keys gespeichert. Der Index wird benötigt, um das von der GPU verschlüsselte Originalpasswort wiederzufinden.

4.3.3 PdfUserKeyMatch-IP

Abbildung 4 zeigt die Struktur der „PdfUserKeyMatch“-IP. Es gibt drei Hauptkomponenten: Input Provider, Crypto Block und Output Handler:

Der Input Provider liest mit Hilfe des PLB Master-Blocks die Encrpytion Keys von einer gegebenen RAM-Adresse, und fügt die Schüssel ins EncKey-FIFO ein. Zusätzlich wird

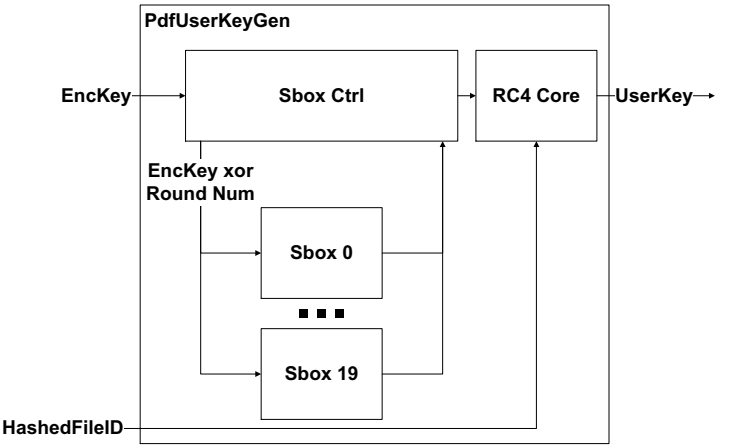

Abbildung 5: Blockdiagramm des User Key-Generators.

der MD5-Hash von der PDF File ID (HashedFileID) vor dem Entschlüsselungsprozess in einem Register gespeichert, welches später vom Crypto Block verwendet wird.

Der Crypto-Block liest die Encryption Keys und die gehashte PDF File ID vom Input Provider und verteilt die Informationen an den User-Key-Generator. Die berechneten User Keys werden zum Output Handler weitergereicht. Abbildung 5 zeigt eine Übersicht über den User-Key-Berechnungsblock. Das Ändern des Encryption Keys (Herstellen des RC4-States) benötigt den größten Berechnungsaufwand mit 1027 Zyklen. Hingegen benötigt die Ver-/Entschlüsselung für eine Runde 68 Zyklen. Deshalb sind die RC4-Zustände als getrennter Block für jede Runde implementiert. Dies erlaubt ein Pipelining des State-Initialisierungsprozesses und die Verwendung eines einzelnen RC4-Blocks für alle Runden.

Der Output Handler vergleicht die berechneten User Keys mit dem von der PDF-Datei erwarteten. Der gefundene Index „Matched UserKey Num“ kann am Schluss über die PLB-Slave-Schnittstelle gelesen werden.

4.3.4 FPGA-Durchsatz

Für die Transformation eines Encryption Keys in den entsprechenden User Key zeigt die Simulation, dass die IP 1360 Taktzyklen für die 20 RC4-Runden benötigt. Die Synthese (Xilinx EDK 14.1) ergibt für das FPGA am ML605-Board, dass 22 IPs mit einer Frequenz von 100 MHz betrieben werden können. Der theoretische Durchsatz beträgt somit 1,6 MKps (Mio. Schüssel pro Sekunde).

5 Cluster-Durchsatz

Die Leistung des gesamten Systems wurde in einem Cluster mit 16 Nodes gemessen. Die Clusterverwaltung wird auf einer separaten Node ausgeführt. Ebenso ist eine mit einer Grafikkarte ausgestattete Node (GPU-Node) im Cluster reserviert. Die restlichen 14 Nodes (FPGA-Nodes) sind jeweils mit einem FPGA-Board bestückt. Jede Node im Cluster ist mit einem 56 Gbit/s FDR-InfiniBand Host Channel Adapter (HCA) ausgestattet.

Der Durchsatz der FPGA-Implementierung wurde ermittelt, indem die Zeit zwischen dem Laden der Encryption Keys in das FPGA DDR RAM und dem Ende der Berechnung am FPGA gemessen wurde. Die beobachteten 0,6 MKps entsprechen jedoch nicht den theoretischen 1,6 MKps. Die Analyse der PLB-Transaktionen mit Xilinx ChipScope ergab, dass das Hauptproblem in der ineffizienten Implementierung der Xilinx PCIe to PLB Bridge liegt. Des Weiteren mindert auch die Verbindung vom PLB zum IP (Xilinx PLB to Intellectual Property Interface Bridge) den Durchsatz. Einen weiteren Grund für die geminderte Leistung stellt der einfache Treiber dar. Dieser bietet weder für Direct Memory Access (DMA) noch für Interrupts Unterstützung. Im Cluster arbeiten alle FPGA-Nodes mit einer Geschwindigkeit von 8,4 MKps.

Für die Durchsatzmessungen der GPU wurden die Encryption Keys berechnet, ohne sie zu den FPGAs zu senden. Die GPU generiert 17,5 MKps.

Das gesamte System berechnet 7,9 MKps, was ungefähr dem FPGA-Durchsatz entspricht. Somit kann gesagt werden, dass bei dieser Anwendung durch das Cluster-Framework keine nennenswerten Einbußen bei der Gesamtperformance des Systems verursacht werden.

Als Vergleich dient eine Implementierungsvariante, die auschließlich mit GPU-Nodes realisiert wurde (siehe Abschnitt 4). Jede GPU-Node ist mit einer NVIDIA GeForce GTX 680 GPU ausgestattet. Da jede GPU den kompletten Algorithmus (inkl. RC4) ausführt, ohne Zwischenergebnisse an andere Coprozessoren weiterzureichen, wird keine Kommunikation benötigt. Der Schlüsselraum muss lediglich auf alle Nodes aufgeteilt werden. Eine einzelne GPU-Node erreicht einen Durchsatz von 680 kKps. 15 GPU-Nodes erzielen somit 10,2 MKps.

6 Ausblick und Verbesserungen

Der Durchsatz der heterogenen Implementierung liegt rund 20 % unter der der homogenen GPU-Implementierung. Der Flaschenhals ist die PCIe-Anbindung des FPGAs. Seit der Veröffentlichung von [DFG+13] wurden die Host-FPGA-Schnittstelle und das FPGA-Design überarbeitet. Für die schnelle Übertragung der Encryption Keys vom Host zum FPGA wurde eine Direct-Memory-Access-Unterstützung (DMA) hinzugefügt. Damit stellt das Laden der Encryption Keys zum FPGA keinen Engpass mehr dar. Im neuen Design steuert der DMA-Controller am FPGA den Datenfluss. Damit können die Encryption Keys ohne den Umweg über das DDR RAM vom PCIe-Block zum „PdfUserMatch“-IP übertragen werden.

Des Weiteren wurde das System auf Interrupts umgestellt, damit die Bandbreite der PCIe-Schnittstelle nicht durch Polling (Prüfen des Done-Bits) belastet wird.

Die Taktfrequenz der Kryptoblöcke kann mit dem Synthese-Tool (Xilinx ISE 14.4) nicht erhöht werden, da der kritsche Pfad im unüberarbeiteten RC4-Block liegt. Aufgrund der Ressourcen-schonenden FPGA-Schnittstelle lassen sich anstatt der 22 RC4-Cores zusätzlich zwei Crypto-Cores instanzieren. Damit ergibt sich ein theoretischer Schüsseldurchsatz von 1,76 MKps. Die Messung auf einer Node ergeben einen Durchsatz von 1,73 MKps. Das überarbeitete Design erreicht somit annähernd eine dreifache Steigerung gegenüber der alten Implementierung.

Ausführliche Tests und Leistungsmessungen im Custer sind noch ausständig. In Summe darf mit diesen Maßnahmen eine signifikate Durchsatzsteigerung für das Gesamtsystem erwartet werden, welche die Leistung der GPU-Variante deutlich übersteigt.

Literatur

[Ado08] Adobe Systems Incorporated. Document management — Portable document format — Part 1: PDF 1.7, 2008.

[DFG+13] B. Danczul, J. Fuss, S. Gradinger, B. Greslehner-Nimmervoll, W. Kastl und F. Wex. Cuteforce Analyzer: A Distributed Bruteforce Attack on PDF Encryption with GPUs and FPGAs. In *Proceedings of the Eighth International Conference on Availability, Reliability and Security (ARES), 2013*, Seiten 720–725. IEEE Computer Society Conference Publishing Services, 2013.

[HMH09] Guang Hu, Jianhua Ma und Benxiong Huang. High Throughput Implementation of MD5 Algorithm on GPU. In *Proceedings of the 4th International Conference on Ubiquitous Information Technologies Applications, 2009. ICUT '09*, Seiten 1–5, 2009.

[KL08] S.H.M. Kwok und E.Y. Lam. Effective Uses of FPGAs for Brute-Force Attack on RC4 Ciphers. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 16(8):1096–1100, 2008.

[LWC+09] Changxin Li, Hongwei Wu, Shifeng Chen, Xiaochao Li und Donghui Guo. Efficient implementation for MD5-RC4 encryption using GPU with CUDA. In *3rd International Conference on Anti-counterfeiting, Security, and Identification in Communication, 2009. ASID 2009*, Seiten 167–170, 2009.

[TL10] Kuen Hung Tsoi und Wayne Luk. Axel: a heterogeneous cluster with FPGAs and GPUs. In *Proceedings of the 18th annual ACM/SIGDA international symposium on Field Programmable Gate arrays*, FPGA '10, Seiten 115–124, New York, NY, USA, 2010. ACM.

[WLT11] Hongwei Wu, Xiangnan Liu und Weibin Tang. A fast GPU-based implementation for MD5 hash reverse. In *Anti-Counterfeiting, Security and Identification (ASID), 2011 IEEE International Conference on*, Seiten 13–16, 2011.

[WYWS12] Feng Wang, Canqun Yang, Qiang Wu und Zhicai Shi. Constant memory optimizations in MD5 Crypt cracking algorithm on GPU-accelerated supercomputer using CUDA. In *2012 7th International Conference on Computer Science Education (ICCSE)*, Seiten 638–642, 2012.