GPU ACCELERATED CIPHERS CRACKING

Jan Schmied
Master Degree Programme (2), FIT BUT
E-mail: xschmi05@stud.fit.vutbr.cz

Supervised by: Vladimír Veselý
E-mail: ivesely@fit.vutbr.cz

Abstract: This paper focuses on one-way hash functions, cryptographic functions and key derivation functions for securing data using password. The work examines time and memory complexity of these algorithms and their potential speedup when run on GPU. It also proposes key features for password cracking on GPU using OpenCL framework.

Keywords: hash functions, cryptography, key derivation functions, GPU, OpenCL

1 ÚVOD

Ochrana dat je v dnešní době velmi důležitá. Abychom ochránili svá data, nejčastěji využíváme šifrování. Existují však legální, ale i nelegální pokusy o prolomení zašifrovaných dat. V některých případech je ovšem získání otevřeného textu zašifrované zprávy téměř nemožné. Například při použití asymetrické kryptografie, nebo symetrické kryptografie s dlouhým klíčem (případně kombinací obou) je prolomení mimo výpočetní sílu dnešních počítačů, ale i těch budoucích, neboť se klíče budou stále prodlužovat. Možnost pro prolomení nám tak dávají pouze aplikace používající krátké uživatelské heslo pro zašifrování. Tato hesla jsou výrazně kratší a mají menší entropii než klíče pro šifrování.

2 HEŠOVACÍ FUNKCE

Přestože se tato práce zabývá prolamováním šifer, jejich prolomení je často závislé na hešovacích funkcích. Všechny dnes běžně používané hešovací algoritmy používají konstrukci Merkle–Damgârd. Ta pracuje tak, že rozdělí vstup na bloky (nejčastěji 512 bitů) a následně je každý blok zpracován konstantním počtem kol. Každé kolo provádí posuny, rotace a logické operace nad blokem, přičemž hlavním prvkem je kompresní funkce, která například z 96bitů udělá 32.

Akcelerace hešovacích funkcí v GPU je velmi efektivní, především proto, že vyžadují velmi málo přístupů do paměti a data se vejdou do registrů. Na CPU dosahujeme rychlosti řádově 10^6 hešu za sekundu, na GPU 10^9 hešů za sekundu. Zrychlení je tedy o tři řády.

3 KRYPTOGRAFICKÉ FUNKCE

Samotné zabezpečení dosahujeme kryptografickými algoritmy, které jsou většinou postaveny na operaci XOR, substituci a dalších matematických operacích jako například násobeni v Galoisově poli v algoritmu AES. V těchto algoritmech se využívá tzv. S-Box, který provádí substituci. Některé algoritmy mají tento S-Box pevně definovaný (DES, AES, ...), jiné ho generují z klíče (Blowfish, RC4, ...).

Právě tyto S-Boxy zapříčiňují velmi malé zrychlení při akceleraci v GPU. Nemohou být umístěny v registrech, protože jsou velké (řádově stovky bajtů), ale musejí být umístěny v pomalejší lokální paměti. Neustálá nutnost načítat tato data zapříčiňují výrazné zpomalení a u algoritmů vyžadujících

modifikaci S-Boxů je zpomalení ještě větší. Proto také šifrovací algoritmy dosahují zrychleni pouze o jeden řád.

4 GENEROVÁNÍ KLÍČŮ

Pokud chceme data zašifrovat s použitím hesla vznikne problém, jak použít krátké heslo s algoritmem, jenž vyžaduje dlouhý klíč, který může opravdu nabývat všech hodnot z prostoru 2^n? Proto existují funkce pro derivaci klíče. V současnosti je použitelná pouze jedna taková standardizovaná(PKCS#5) funkce. Jedná se o funkci PBKDF2, která iterativně hešuje vstup funkcemi HMAC a vyžaduje také salt, který zabrání vytvoření rainbow table. Tato velmi jednoduchá funkce dokáže snížit rychlost o libovolný počet řádů v závislosti na počtu iterací. Minimální doporučený počet iterací je 1000, které sníží rychlost o tři řády.

Další zajímavé funkce jsou bcrypt a scrypt, které cílí přímo na zpomalení výpočtu na GPU, ale nejsou standardizované.

5 APLIKACE KRYPTOGRAFICKÝCH PROSTŘEDKŮ

Mnoho aplikací umožňuje uživateli zadat heslo a data zašifrovat. Zaměřme se ale na tři často používané formáty dat. Jsou to PDF, ZIP a DOC. Všechny tyto formáty umožňují šifrování bezpečnými algoritmy. Pro brute-force útok (ale i pro korektní aplikaci) je ovšem důležité jak zjistit, že zvolené heslo je správné? K tomu slouží ověřovací hodnota, která většinou vzniká při generování šifrovacího klíče, nebo následnými operacemi s klíčem.

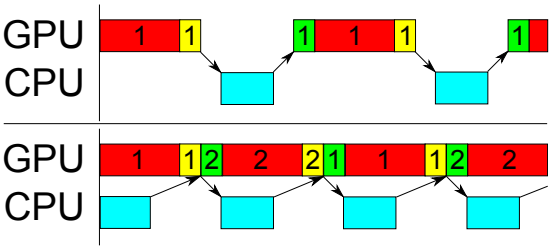
PDF používá ověřovací hodnotu délky 32 bajtů vytvořenou použitím algoritmů MD5+RC4 a DOC délky 16 bajtů z algoritmů RC4+MD5. U těchto formátů je nemožné dostat false positive heslo. V případě formátu ZIP je využita dvou bajtová hodnota vzniklá při vytváření klíče funkcí PBKDF2 a proto false positive hesla vznikají velmi často. Ty je nutné ověřit tak, že spočítáme HMAC-SHA1 kontrolní součet zašifrovaných dat s klíčem získaným opět z PBKDF2.

6 AKCELERACE POMOCÍ GPU

Grafické procesory poskytují velmi vysoké zrychlení paralelizovatelných úloh, protože obsahují mnoho (až tisíce) velmi jednoduchých jader. Pro brute-force útoky na hesla jsou nejvhodnější GPU od firmy AMD(ATI), protože každé jádro obsahuje ALU, na rozdíl od GPU Nvidie, kdy více jader sdílí jedno rychlé ALU. Pro karty od AMD je potřeba použít framework OpenCL.

Základní princip akcelerace pomocí OpenCL spočívá v nahrání velkého množství dat do zařízení a nad těmito daty se spustí kernely, které běží na GPU. Po dokončení výpočtu je nutné výsledky nahrát zpět do operační paměti a analyzovat. Zde vzniká problém s rychlostí rozhraní PCI-Express, ke kterému jsou grafické karty připojeny. To ve verzi 3.0 poskytuje rychlost 16GB/s, která je ovšem vzhledem k výpočetní rychlosti GPU velmi nízká. Současná GPU umožňuje rychlost výpočtu až 9Gh/s funkce MD5. Proto je nutné eliminovat přenosy mezi operační pamětí a zařízením. Další problém je schopnost generovat takový počet hesel v CPU. Generátor hesel proto musí být umístěn přímo v GPU. Toto umožňuje současně efektivně využít více GPU s různou rychlostí.

V jednoduchém případě se na GPU spustí kernel a ten zpracuje data. Výsledky se nahrají zpět do CPU a zkontroluje se případná nalezená shoda. Nyní ovšem GPU neprovádí výpočty a tak není plně využit jeho potenciál. Tento nedostatek je možné odstranit zařazením více úloh do fronty. Nyní zatímco CPU ověřuje shodu předchozího balíku dat, GPU už počítá s novými daty.



Obrázek 1: Jednoduché a vylepšené řazení úloh

7 VÝSLEDKY AKCELERACE

Navrhovaná vylepšení byla implementována na GPU AMD Radeon HD 7970 a CPU Intel Celeron G465.

Algoritmus (global work size)	CPU baseline	Generátor (jednoduché řazení)		Řazení (generátor v GPU)	
		v CPU	v GPU	jednoduché	vylepšené
Raw-MD5 (2^{22})	4,5Mh/s	35Mh/s	3,0Gh/s	3,0Gh/s	3,0Gh/s
Raw-MD5 (2^{19})	4,5Mh/s	31Mh/s	2,6Gh/s	2,6Gh/s	3,0Gh/s
PDF-MD5+RC4 (2^{17})	42kh/s	1,5Mh/s	1,7Mh/s	1,7Mh/s	1,7Mh/s
PDF-MD5+RC4 (2^{13})	42kh/s	900kh/s	1,0Mh/s	1,0Mh/s	1,7Mh/s

Tabulka 1: Rychlosti výpočtu s vylepšeními

Z naměřených výsledků vyplývá, že obě navrhovaná vylepšení mají vliv na výkonnost, ale jejich efektivita je závislá na dalších faktorech. Generátor v GPU značně zvýší rychlost pro velmi rychlé výpočty, ale pro pomalé je zrychlení pouze v řádech procent. Vylepšené řazení přináší zrychlení, pouze pokud je global work size menší než optimální hodnota pro danou úlohu. Proto je vhodné pouze pro starší GPU, které nemohou nastavit větší global work size, protože mají například málo paměti.

Generátor v GPU se používá velmi často v lamačích hešů, ale vylepšené řazení se nikde nepoužívá, protože jeho přínos u moderních GPU je sporadický.

8 ZÁVĚR

Prolamování šifer, hešů a hesel je velmi časově náročné, a proto je akcelerace pomocí GPU velmi vhodná. Existují však také funkce zaměřené na zpomalení brute-force útoku obecně a pokročilejší cílí přímo na snížení efektivity výpočtu na GPU. Běžně používané aplikace, které umožňují zabezpečení heslem, obsahují ověřovací hodnotu, která zpříjemňuje útok. Kdyby tato hodnota neexistovala, bylo by velmi obtížné ověřit správnost testovaného hesla. Pro co nejefektivnější využití GPU je nutné generovat hesla přímo v zařízení a minimalizovat čekaní zařízení mezi výpočty nastavením vhodné global work size.

V rámci diplomové práce se zabývám implementací aplikace, která umožní akcelerovat brute-force prolamování hesel pomocí GPU. Podporované formáty pro crackování jsou PDF, DOC a ZIP, ale je možné jednoduše doplnit další formáty.

REFERENCE

[1] MENEZES, Alfred J., Paul C. VAN OORSCHOT a Scott A. VANSTONE. Handbook of applied cryptography. Vyd. 1. Boca Raton: CRC Press, 1997, 780 s. ISBN 08-493-8523-7.

[2] ADVANCED MICRO DEVICES. AMD Accelerated Parallel Processing: OpenCL Programming Guide [online]. 2013 [cit. 2014-02-20].