TRUECRACK

Il software “TRUECRACK” effettua un attacco a forza bruta per trovare la password di un volume TrueCrypt© cifrato con le impostazioni di default del programma. Per diminuire il tempo previsto per gli attacchi, tale software è stato ottimizzato sfruttando l’infrastruttura di programmazione proposta da nVidia in CUDA.

Questo software è stato studiato e realizzato come progetto del corso di Architetture e Algoritmi per Sistemi Crittografici del Politecnico di Milano.

Gli autori di questo software, Luca Vaccaro e Riccardo Zucchinali, frequentano il secondo anno di Laurea Specialistica presso la facoltà di Ingegneria Informatica del Politecnico di Milano.

Il software è basato sul codice sorgente di TrueCrypt© (versione di riferimento 7.0).

INTRODUZIONE

TrueCrypt© è un noto programma applicativo usato per la crittazione *on-the-fly* di interi dischi rigidi o loro partizioni (OTFE/On-the-fly-Encryption). L'intero disco virtuale crittografato viene salvato in un unico file che può essere montato come un disco fisico.

Il software si propone come breaker di password per files di volume TrueCrypt, in altre parole è in grado di forzare passphrases di lunghezza arbitraria nelle seguenti modalità:

- partire da un dizionario disponibile sotto forma di testo contenente una parola per riga
- generando tutte le possibili combinazioni dato un alfabeto di possibili simboli contenuti nella password.

Nel documento si inizia a descrivere l’intera procedura di crittografia utilizzata da TrueCrypt© nel caso di selezione dei parametri standard:

- funzione di derivazione delle chiavi: ***PBKDF2*** basato su ***RIPEMD160***.
- algoritmo di cifratura dati su disco: ***XTS*** basato su ***AES***.

Successivamente si descrive la procedura di decifratura, o meglio il processo operativo che il software Truecrack utilizza per ricavare e verificare la password corretta dal volume cifrato. Questo processo complesso e costoso in termini computazionali è ottimizzato per mezzo dell’infrastruttura Cuda di nVidia. Attraverso questa tecnologia si potrà ottenere un notevole speedup e ridurre notevolmente il tempo impiegato dagli attacchi.

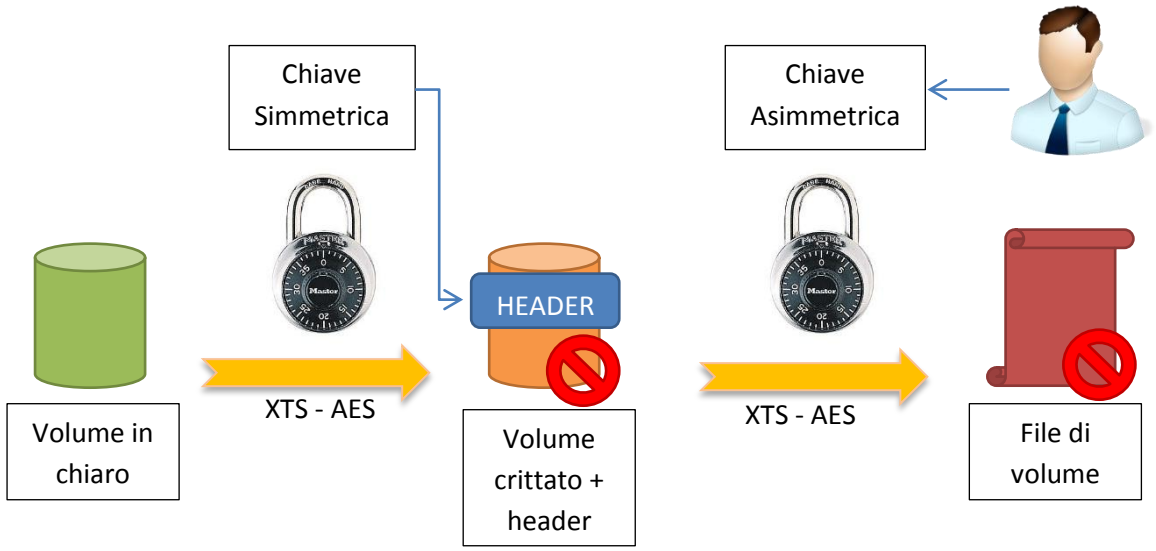
Infine si fornisce una breve descrizione dei parametri in fase di compilazione ed esecuzione del programma, nonché una valutazione delle performance.

TRUECRYPT

La procedura di crittografia del generico volume TrueCrypt si compone di due fasi principali:

1. Crittografia simmetrica dei dati del volume tramite “Master Key”.
2. Crittografia asimmetrica dell’intestazione del volume tramite “Header Key”.

La procedura di crittografia dei dati utilizza in entrambi i casi il protocollo ***XTS*** come modalità operativa del cifrario usate per la crittografia dei dischi. Gli algoritmi di cifratura supportati da TrueCrypt sono: ***AES***, ***Serpent***, and ***Twofish***. In questa documentazione si analizza solo il la cifratura basata su ***AES***. Successivamente verrà riportata la procedura di derivazione della Header key. Nella figura sottostante è possibile osservare il procedimento di crittazione.



Crittografia simmetrica

Nella prima fase, il volume in chiaro viene crittografato per mezzo di una chiave simmetrica “Master Key” secondo il protocollo ***XTS*** che a sua volta si appoggia al protocollo ***AES***. La chiave simmetrica è generata dal generatore di numeri pseudocasuali, interno a truecrypt, durante la fase creazione del volume. Tale chiave viene memorizzata in chiaro nell’area header di intestazione del file crittato; nella sezione di intestazione sono contenute varie informazioni, quali: i protocolli utilizzati, la password simmetrica, i flags di controllo, la versione del software utilizzato, etc..

Crittografia asimmetrica

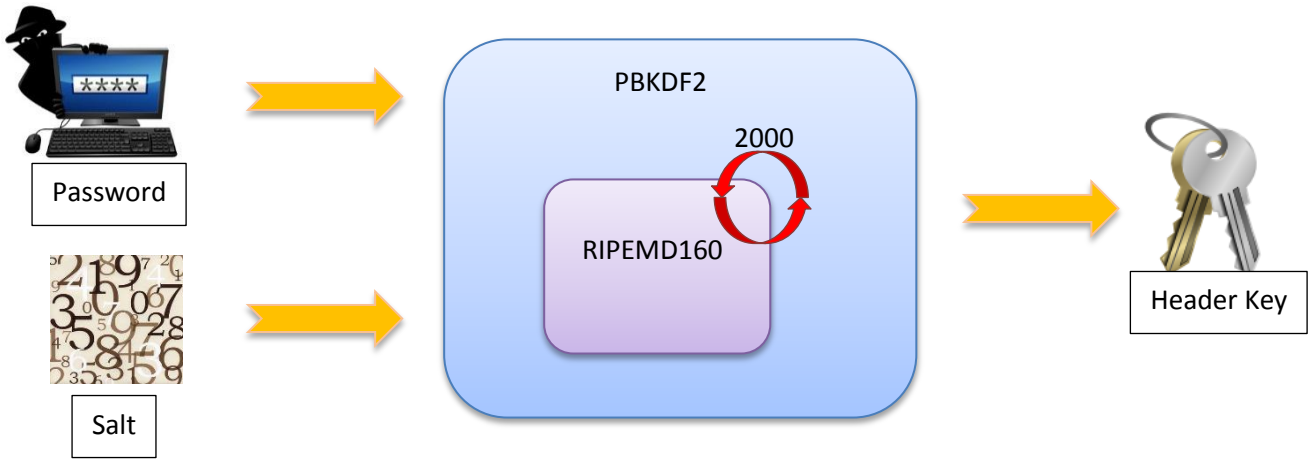
Al volume precedentemente crittato viene anteposta la sezione di header, contente in chiaro le informazioni utili per la decriptazione e a scopi informativi; queste informazioni sono visibili in chiaro ed è quindi possibile recuperare direttamente la “Master Key”. Per questo motivo la sezione di header viene criptata a sua volta usando il protocollo ***XTS*** – ***AES*** tramite una chiave asimmetrica, chiamata “Header Key”.

Generazione della chiave Header Key

La chiave “Header key” viene generata al momento di creazione del volume secondo una procedura complessa ed articolata per evitare o ritardare eventuali attacchi di tipo password cracking.

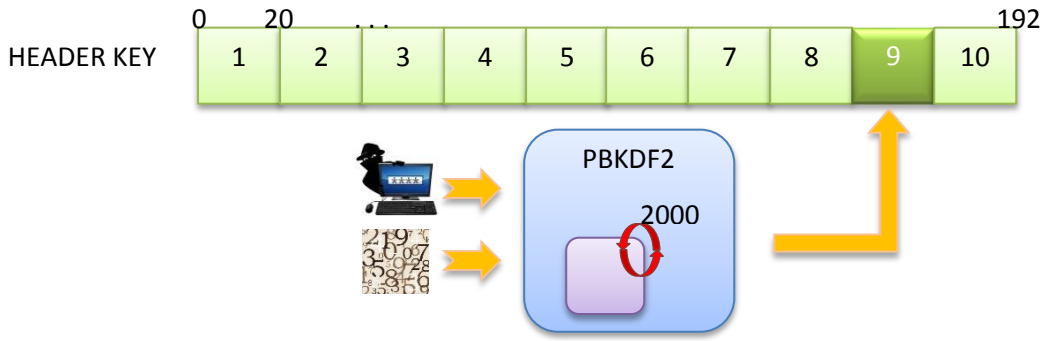
Per aumentare la sicurezza alla password si associa il salt, una stringa esadecimale di lunghezza 64bytes associata alla password, che riduce la possibilità di sfruttare un dizionario precomputato di password; il salt è generato in modo casuale in fase di creazione del volume ed è salvato nelle prima 64 posizioni del file di volume truecrypt.

L’algoritmo ***PBKDF2*** (Password-Based Key Derivation Function) è una funzione di derivazione delle chiavi che fa parte dello standard di RSA Laboratories’ Public-Key Cryptography Standard, PKCS #5 v2.0. L’algoritmo riceve in ingresso una password o passphrase e un salt e applica una funzione pseudo casuale, come cryptographic hash, cipher o HMAC, molte volte fino a produrre la Derived Key usata come chiave asimmetrica Header Key. L’implementazione di truecrypt supporta le seguenti procedure di cryptographic hash: ***RIPEMD160***, ***SHA-512*** e ***Whirpool***. In questo progetto si considera la procedura ***RIPEMD160*** che viene iterata 2000 volta secondo l’algoritmo ***PBKDF2***.



Il ***RIPEMD*** è un algoritmo crittografico di hashing ideato nella università belga Katholieke Universiteit Leuven e fu pubblicato nel 1994. Il ***RIPEMD*** nacque come alternativa europea ad altre funzioni di hash di provenienza americana come l’***MD4*** e l’***MD5***. Il message disgest di ***RIPEMD160*** è costituito da 160-bit che producono un hash di 20 byte, quindi 40 simboli esadecimali.

Per questo motivo la “Header Key” di 192byte viene suddivisa in 10 blocchi da 20 byte ciascuno, a parte l’ultimo di 12bytes; su ogni blocco viene effettuato l’algoritmo ***PBKDF2*** che itera la procedura di ***RIPEMD160*** 2000 volte.



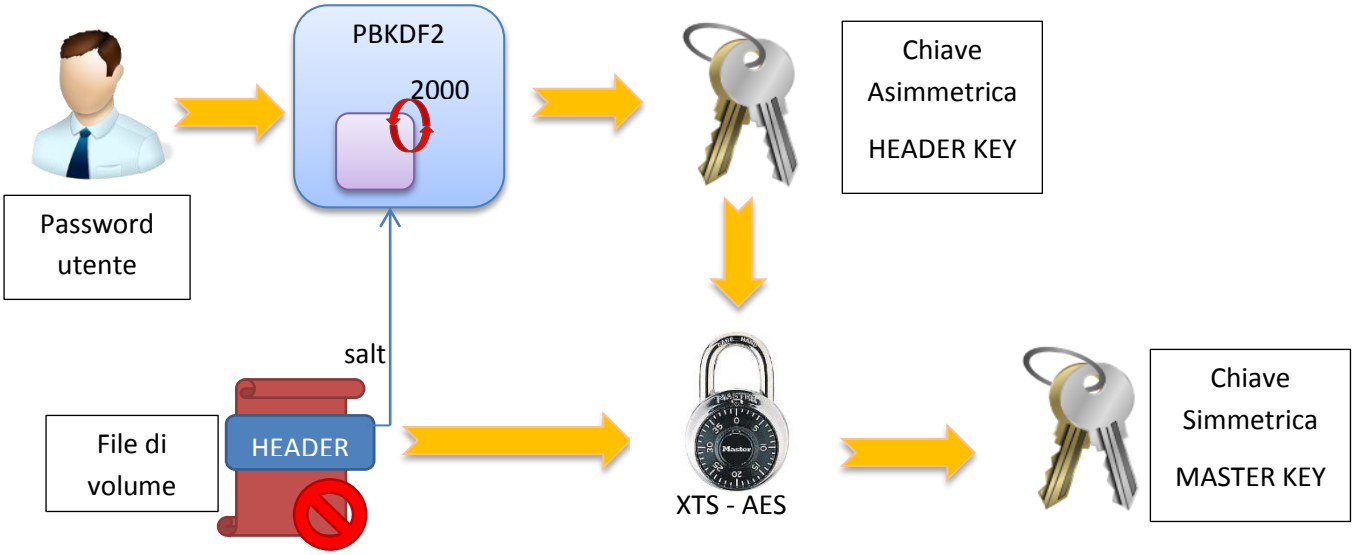
DECIFRAZIONE

La procedura di decifrazione implementata nel software Truecrack ha come obiettivo ricavare la MASTER KEY a partire da una generica password utente e dal volume cifrato.

In particolare saranno interessati i primi 64 bytes del file di volume truecrypt che rappresentano il valore esadecimale del salt da associare alla password; mentre la password è generata di volta in volta tramite dizionario o combinazione di alfabeto. La procedura ***PBKDF2*** genera l'hash della HEADER KEY a partire dal salt e dalla password. Successivamente si decifra l'intestazione HEADER del volume tramite la chiave asimmetrica precedente trovata HEADER KEY. Se la chiave risulta corretta è possibile recuperare la MASTER KEY a partire dal 256-esimo byte.

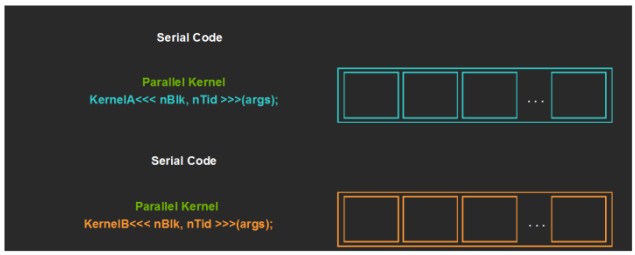
Per verificare il successo o l'insuccesso dell'operazione di decifrazione sono presenti alcuni flags di controllo nella sezione decifrata di HEADER:

- 64-68 bytes: Ascii string "TRUE"
- 72-76 bytes: CRC-32 checksum dei bytes decifrati 256-511
- 252-256 bytes: CRC-32 checksum dei bytes decifrati 64-251
-



NVIDIA CUDA

CUDA è un'architettura di elaborazione in parallelo realizzata da NVIDIA che permette netti aumenti delle prestazioni di computing grazie allo sfruttamento della potenza di calcolo delle GPU (unità di elaborazione grafica). In fase di test è stata utilizzata una macchina equipaggiata con una Geforce GTX470 e il toolkit CUDA preinstallato.

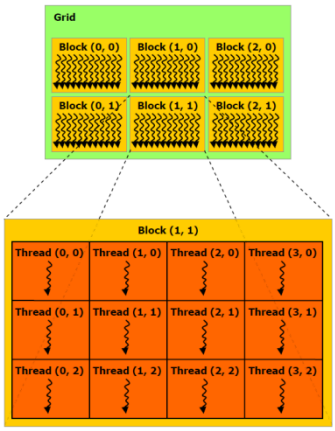


Applicazione

Un'applicazione CUDA è composta da parti seriali, normalmente eseguite dalla CPU di sistema (host) e da parti parallele (kernel) eseguite dalla GPU, o nei termini usati da nVidia, dal device.

Parallelismo

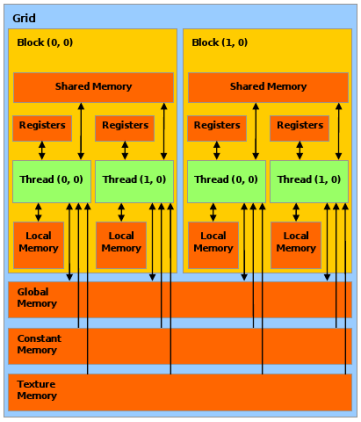
Un kernel è definito come un griglia (grid) che può essere a sua volta decomposta in blocchi (blocks), che vengono assegnati ai vari multiprocessori e rappresentano il parallelismo a grana grossa. All'interno dei blocks c'è l'unità di computazione fondamentale il thread ad una granularità di parallelismo molto fine. Un thread può appartenere ad un solo blocco ed è identificato da un indice univoco per tutto il kernel. Maggiore è il numero di threads in parallelo, migliori sono le performance.



Tipologie di memorie

L'architettura Cuda dispone delle seguenti tipologie di memoria:

- Global : variabili condivise da tutti i blocchi e da tutti i threads accessibili dal sistema host
- Constant: variabili delle costanti comune a tutti i blocks
- Local : variabili locali interne diverse per ogni threads
- Shared : variabili condivise dai threads di uno stesso block
- Texture : matrici di variabili destinate alla grafica



Qualificatori di funzione

Un kernel CUDA è una particolare funzione C che, invocata dall'host (CPU), viene eseguita sul device (GPU). I kernel hanno caratteristiche ben precise : tipo void di ritorno, non essere ricorsive, non possono avere numero di parametri variabili e nemmeno variabili di tipo statico. In CUDA le funzioni facenti parti di un kernel vanno identificate tramite appositi qualificatori da anteporre alla loro dichiarazione, quali:

- __global__ : identifica la funzione principale del kernel invocata dal codice dell'host,
- __device__ : identifica una funzione chiamata dal codice che gira sulla GPU
- __host__ : è una funzione chiamata dall'host e viene eseguita sull'host

L'invocazione di un kernel avviene all'interno del codice eseguito sulla CPU, chiamandone il nome e specificando la configurazione di esecuzione all'interno di appositi identificatori (<<< e >>>) con la sintassi:

```
Kernel <<< dim3 grid, dim3 block >>> ( parameter 1, parameter2, ... );
```

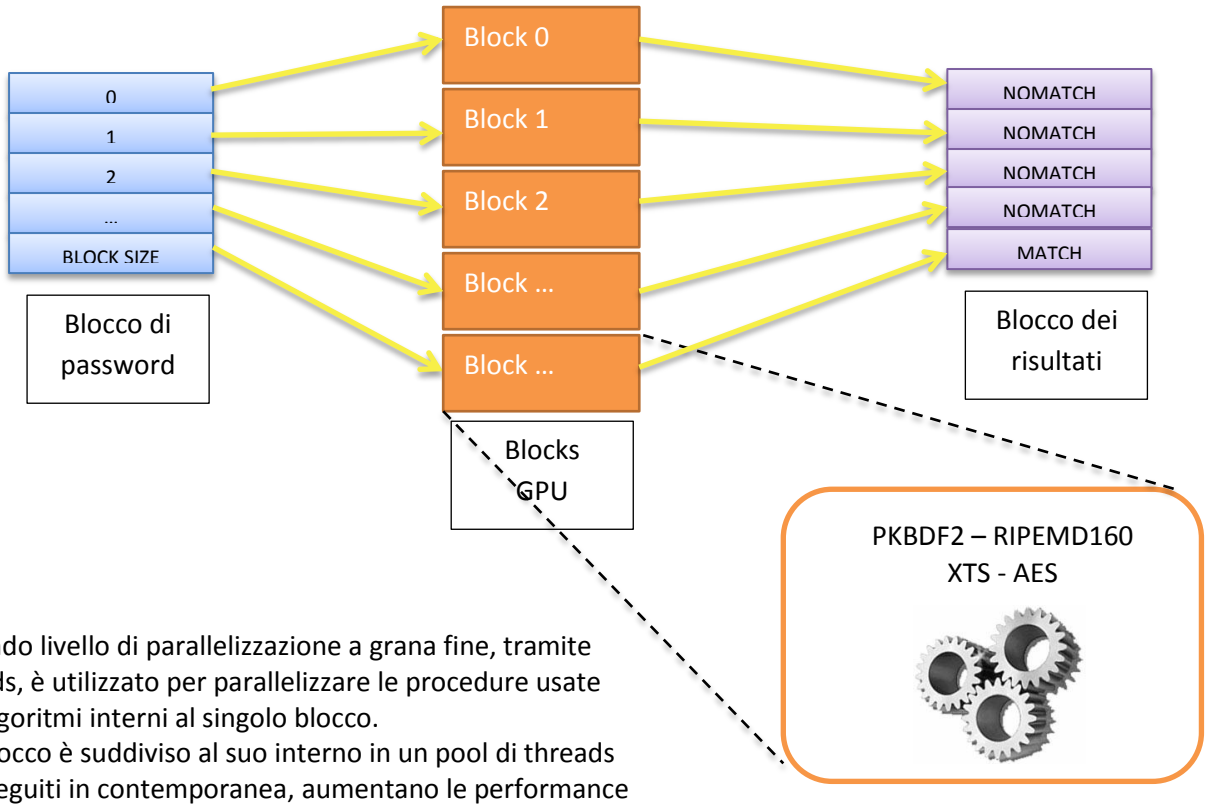
Dove grid e block sono delle variabili che specificano il numero di blocchi in cui è diviso il kernel e il numero di threads all'interno di ciascun blocco.

PARALLELIZZAZIONE

Gli attacchi bruteforce, sia basati su dizionario che su combinazione di alfabeto, richiedono di effettuare una vasta gamma di tentativi fino a trovare la password corretta. I crack tradizionali su CPU effettuano in ordine sequenziale un numero elevato di prove, dove per prova si intende l'esecuzione dell'intero processo di verifica a partire dalla password.

TrueCrack sfrutta l'infrastruttura Cuda per effettuare il calcolo e la verifica di più password in parallelo. Per ottenere migliori performance sono stati parallelizzati gli algoritmi interni laddove possibile.

In altre parole il primo livello di parallelizzazione a grana grossa, tramite i blocks, permette di eseguire l'intero processo di verifica su più password in contemporanea; ogni password usa un block parallelo di GPU.



Il secondo livello di parallelizzazione a grana fine, tramite i threads, è utilizzato per parallelizzare le procedure usate dagli algoritmi interni al singolo blocco.
Ogni blocco è suddiviso al suo interno in un pool di threads che, eseguiti in contemporanea, aumentano le performance degli algoritmi.
Gli algoritmi sono parallelizzati in threads nella seguente configurazione:

- 10 threads per PBKDF2
- 1 threads per XTS

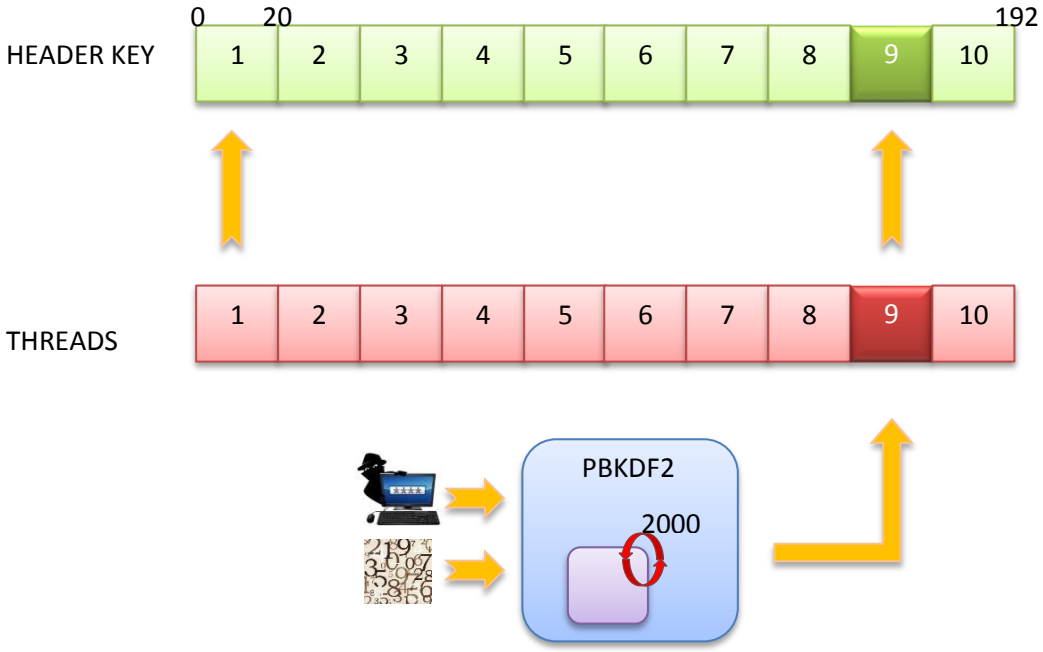
PBKDF2 parallelizzato

Come visto precedentemente la procedura di generazione della chiave sfrutta l'algoritmo ***PBKDF2***. Tale procedura calcola l'Header Key a blocchi di 20 caratteri esadecimali alla volta tra loro indipendenti. Detto questo, TrueCrack utilizza 10 threads in parallelo per calcolare l'hash dell'Header Key .

XTS parallelizzato

L'algoritmo ***XTS*** suddivide lo stream di dati in macro-blocchi di 512 bytes tra loro indipendenti. Ogni macro-blocco si suddivide in altri 32 blocchi da 16 bytes tra loro dipendenti. Su ogni blocco viene poi eseguito il calcolo dell'hash tramite ***AES***. Considerato che i 32 blocchi interni sono tra loro legati e dipendenti, è stata

effettuata una parallelizzazione a livello di macro-blocco; tuttavia basta un solo macro-blocco perché l'header è composto da esattamente 512 bytes. Per questo motivo risulta sufficiente l'uso di un solo thread per block.



CODICE SORGENTE

In questa sezione si sono estrapolati alcuni passaggi chiavi del codice sorgente, in modo da spiegare meglio le ragioni che hanno portato a questa implementazione. In particolare si citano e descrivono i passaggi legati all'infrastruttura Cuda.

Configurazione Cuda

La funzione Kernel, eseguita sulla GPU, effettua l'intero processo di generazione della Header Key e decifratura del volume, prendendo come ingresso: il salt, l'header criptato, il blocco di password. Il blocco di password *dev_blockPwd* è costituito da un array monodimensionale che contiene sequenzialmente le password da testare; per indirizzare efficacemente le singole password all'interno dell'array si utilizzano i vettori *dev_blockPwd_init* e *dev_blockPwd_length* che rappresentano rispettivamente la posizione iniziale e la lunghezza di ogni password nell'array. Nell'architettura CUDA è infatti sconsigliato l'utilizzo di puntatori multipli e passaggio di strutture a matrici. L'invocazione della funzione kernel è la seguente:

```
cuda_Kernel <<< blockGridSizeCurrent, 10 >>>  (dev_salt, dev_header,
dev_blockPwd, dev_blockPwd_init, dev_blockPwd_length, dev_result);
```

Si osservi che l'ultimo parametro è un vettore di interi che, per ogni password provata, contiene la macro *MATCH* o *NOMATCH* rispettivamente se la password è quella corretta oppure no.

Funzione Kernel

La funzione *cuda_Kernel* rappresenta il processo di verifica della password. L'esecuzione di *cuda_Kernel* per mezzo della GPU genera un pool di blocks e di threads in parallelo: ad ogni block è associata una password ed ad ogni thread è associato un blocco operazionale di computazione per gli algoritmi. Per identificare il block e il thread corrente in esecuzione si usano le direttive *blockIdx.x* e *threadIdx.x.*

Una volta identificata la password da verificare, il kernel richiama la funzione di generazione della Header Key, *cuda_Pbkdf2()*. L'hash è calcolato da 10 threads che si eseguono in parallelo e ogni thread effettua l'hash per il blocco di 20 caratteri associato al thread. La Header Key è memorizzata in memoria condivisa in modo tale che è possibile ottenere l'intera stringa una volta sincronizzati i vari threads dello stesso blocco.

Successivamente sulla Header Key cosi' trovata si procede con la decifratura dell'header, tramite la funzione *cuda_Xts()*. Nel caso in cui la password sia corretta la funzione ritorna con la macro *SUCCESS.* Nell'array dei risultati viene salvato l'eventuale successo o insuccesso, con le macro *MATCH* o *NOMATCH*, in posizione della password provata.

```
__global__ void cuda_Kernel ( unsigned char *salt, unsigned char
*headerEncrypted, unsigned char *blockPwd, int *blockPwd_init, int
*blockPwd_length, short int *result) {

    int numData=blockIdx.x;
    int numBlock=threadIdx.x;

    // Array of unsigned char in the shared memory
    __shared__ __align__(8) unsigned char headerkey[192];
    __shared__ __align__(8) unsigned char headerDecrypted[512];

    // Calculate the hash header key
    cuda_Pbkdf2 (salt, blockPwd, blockPwd_init, blockPwd_length,
headerkey, numData, numBlock);

    // Synchronize all threads in the block
    __syncthreads();

    // Decrypt the header and compare the key
    if (numBlock==0) {
        int value;
        value=cuda_Xts (headerEncrypted, headerkey, headerDecrypted);

        if (value==SUCCESS)
            result[numData]=MATCH;
        else
            result[numData]=NOMATCH;
    }
}
```

COMPILAZIONE

Il software supporta due differenti modalità di utilizzo:

- standard tramite cpu
- optimized tramite cuda

Per selezionare una delle due differenti modalità basta definire in fase di compilazione rispettivamente la variabile ***GPU*** a ***false*** o ***true***. Nel caso si vuole la modalità ottimizzata usare il comando di compilazione:

```
make GPU=true
```

ESECUZIONE

Il software Truecrack funziona da linea di comando. Se non vengono passati parametri o si richiama help è possibile visualizzare il menù con le possibili opzioni di esecuzione:

```
Bruteforce password cracker for Truecrypt volume.
Optimized with Nvidia Cuda technology.
Based on TrueCrypt, freely available at http://www.truecrypt.org/
Copyright (c) 2011 by L. Vaccaro & R. Zucchinali
Usage: ./bruteforce options [ inputfile | value ] volumefile
 -h --help            Display this usage information.
 -t --truecrypt FILE  Truecrypt volume file.
 -w --wordlist FILE   Wordlist mode, read words from FILE.
 -m --maxlength INT   Charset mode, max length of words generated.
 -c --charset STRING  Charset mode, create words from charset STRING.
 -b --blocksize INT   Block size of words parallel computed.
 -v --verbose         Show cracked passwords.
```

Nel dettaglio le opzioni sono le seguenti:

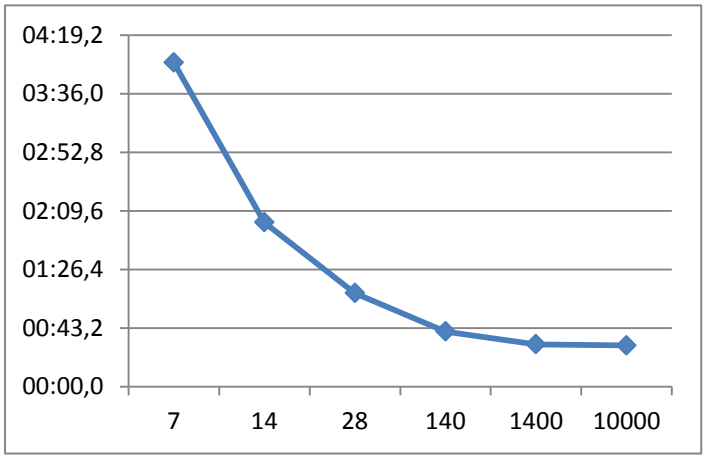
- TrueCrypt volume : Il volume Truecrypt su cui effetturare l'attacco.
- Wordlist FILE : Il file dizionario contenente una lista di parole, una per riga.
- MaxLength INT : La dimensione massima delle parole generate tramite il charset.
- Charset STRING : L'alfabeto dal quale partire per generare le possibili password. Il software effettua la permutazione dei caratteri generando l'intero set di combinazioni a partire da un carattere per volta fino a passphrases di dimensione *MaxLength*.
- Blocksize INT : Dimensione del blocco di parole da computare in parallelo, di default 1024. Al variare di questo parametro, varia la memoria dinamica occupata dal processo e il numero di istanze, blocchi, in Cuda eseguiti in parallelo. Quest'ultimo ha come effetto un significativo mutamento delle performance, ed in particolare maggiori sono i blocchi Cuda in parallelo e minore sarà il tempo di attacco del brutefroce. Il valore di default è il numero di unità processori di cui è equipaggiata la scheda.
- Verbose : Stampa volta in volta le password provate e scrivendo in output il relativo messaggio di successo od insuccesso di un tentativo.

VALUTAZIONE E PERFORMANCE

Il livello di parallelizzazione, tramite l'infrastruttura Cuda, ha prodotto un notevole incremento delle performance dell'attacco. Le prove sono state effettuate utilizzando la scheda Nvidia GeForce GTX 470.

SPECIFICHE DEL MOTORE DELLA GPU:	
CUDA Cores	448
Clock grafico (MHz)	607
Clock del processore (MHz)	1215
Fill Rate texture (miliardi/s.)	34

Nella figura1 si trovano i tempi di esecuzione GPU impiegati per un attacco di tipo dizionario ad un volume TrueCrypt con un file di 10'000 parole di lunghezza media 10 caratteri. Si noti il carattere decrescente della curva dei tempi di esecuzione in GPU al crescere di numeri di blocchi paralleli: un blocco equivale al calcolo e verifica di una password. Lo stessa configurazione su CPU impiega mediamente 11m e 1,1 secondi su macchina Linux: Intel(R) Core(TM) i7 920 a 2.67GHz.

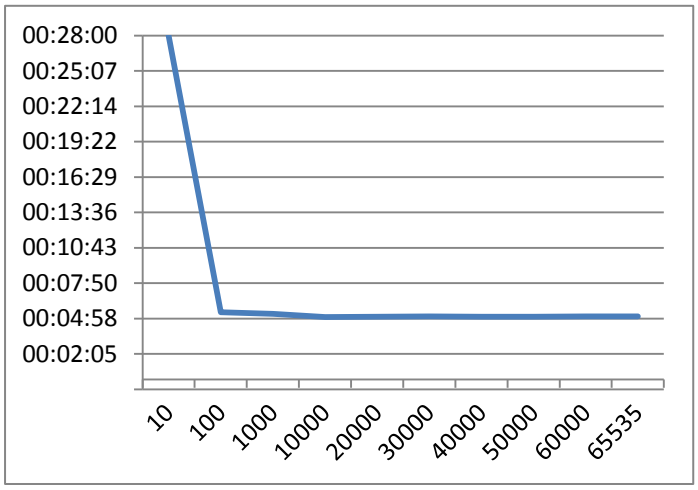


Parallel Blocks	GPU
7	3m 58,919s
14	2m 1,170s
28	1m 8,915s
140	0m 40,691s
1'400	0m 31,234s
10'000	0m 30,425s

Si sono presi i campioni multipli del numero di multiprocessori di cui è equipaggiata la scheda GTX 470 (14)

Figura 1: Tempi di esecuzione di attacco a dizionario con file di 10'000 parole su CPU e GPU parallelizzato su più blocchi.

Nella figura 2 si mostra il rapporto tra il tempo totale di esecuzione ed il numero di blocchi eseguiti in parallelo nel caso di un insieme più numeroso di tentativi: un file di 100'000 parole. Si è arrivati a calcolare fino a 65'535 blocchi in parallelo perché è il limite massimo di grandezza di una dimensione della griglia dei blocks. Si osservi come i tempi di esecuzione si stabilizzano su un valore fisso oltre il quale la parallelizzazione non porta più benefici.



Parallel Blocks	Tempi GPU
10	27m 59,839s
100	5m 27,976s
1'000	5m 19,936s
10'000	5m 4,465s
20'000	5m 6,977s
30'000	5m 7,319s
40'000	5m 6,353s
50'000	5m 5,629s
60'000	5m 7,540s
65'535	5m 7,200s

Figura 2: Tempi di esecuzione di attacco a dizionario con file di 100'000 parole parallelizzato su più blocchi.

CURIOSITA'

" In July 2008, several TrueCrypt-secured hard drives were seized from Daniel Dantas, who was suspected of financial crimes. The Brazilian National Institute of Criminology (INC) tried for five months (without success) to obtain access to TrueCrypt-protected disks owned by the banker, after which they enlisted the help of the FBI. The FBI used dictionary attacks against Dantas' disks for over 12 months, but were still unable to decrypt them. "

Fonti:

- FBI hackers fail to crack TrueCrypt [http://news.techworld.com/security/3228701/fbi-hackers-fail-to-crack-truecrypt/]
- Wikipedia Daniel Dantas [http://en.wikipedia.org/wiki/Daniel_Dantas]

BIBLIOGRAFIA

- CUDA by Example: An Introduction to General-Purpose GPU Programming: http://developer.nvidia.com/cuda-example-introduction-general-purpose-gpu-programming
- NVIDIA CUDA programming guide: http://developer.download.nvidia.com/compute/cuda/2_0/docs/NVIDIA_CUDA_Programming_Guide_2.0.pdf
- TrueCrypt documentations: http://www.truecrypt.org/docs/
- TrueCrypt User Guide: http://www.sci-lib.net/index.php?act=attach&type=post&id=10727
- PKCS #5 v2.0: Password-Based Cryptography Standard, RSA Data Security, Inc. Public-Key Cryptography Standards (PKCS), March 25, 1999: ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-5v2/pkcs5v2-0.pdf
- Nvidia Geforce GTX470 Datasheet: http://www.nvidia.com/docs/IO/90025/GTX-480-470-Web-Datasheet-Final4.pdf