High Performance Computing nella crittoanalisi

Marco Cianfriglia* Gruppo di ricerca CRANIC @IAC-CNR, Roma

Istituto per le Applicazioni del Calcolo
"Mauro Picone"
Consiglio Nazionale delle Ricerche
Via dei Taurini 19, Roma
*m.cianfriglia@iac.cnr.it
Credits to: E. Agostini, M. Bernaschi, S. Guarino

Outline

Introduzione

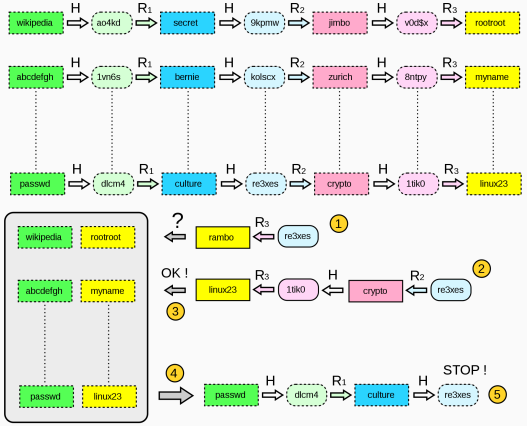
GPGPU - General Purpuse computing on Graphics Processing Units

- GPU nate per accelerare la grafica
- oggi utilizzate per machine learning, computer vision, manipolazione delle immagini, fisica, chimica, finanza, biologia, analisi dati e segnali, ...
- centinaia applicazioni sviluppate e supportate da numerosi fornitori[1]
- possono velocizzare l'esecuzione di parti critiche delle applicazioni
- particolarmente adatte per operazioni semplici da ripetere su grandi moli di dati (SIMD[2])

[1] `http://www.nvidia.com/object/cuda_showcase_html.html`
[2] Single Instruction Multiple Data

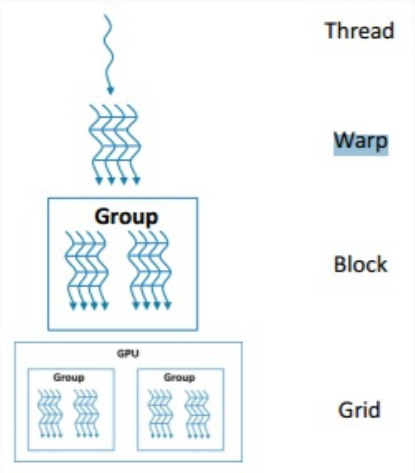
Parallelizzare Attacchi a "Forza Bruta"? Non solo!

Dictionary/Time-Memory Trade-Off: testare le chiavi di un dato insieme ottimizzando le risorse disponibili



Velocizzare gli attacchi usando GPU è non banale:

- l'organizzazione delle unità di calcolo pone vincoli di elaborazione e memoria
- capire come/dove intervenire per massimizzare il vantaggio è application-dependent

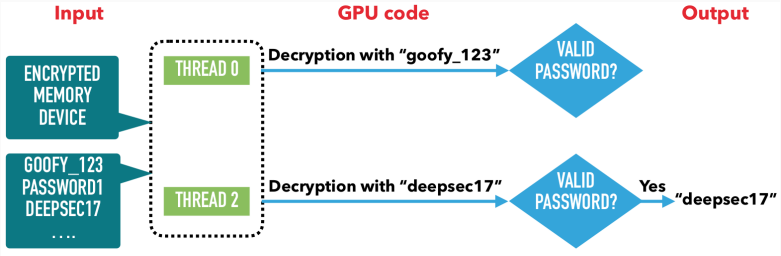


Dictionary Attack su Sistemi Crittografici Specifici

Parallelizzare un Dictionary attack su GPU

In un dictionary attack, l'operazione da eseguire è tipicamente identica e si può applicare il paradigma Single Program, Multiple Data (SPMD):

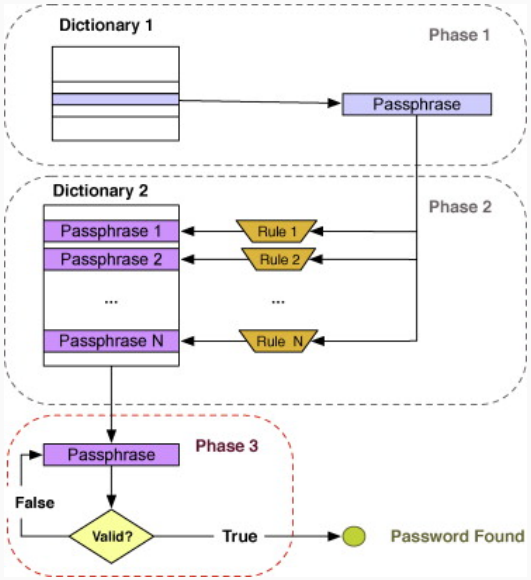
- tutti i thread eseguono le stesse istruzioni su input diversi



Problemi e soluzioni:

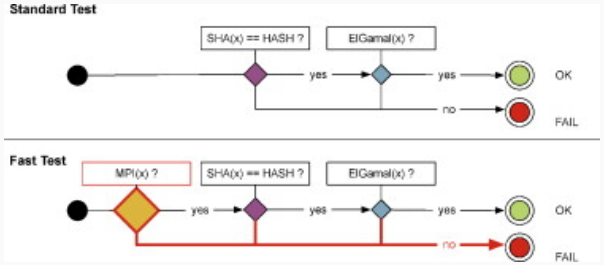
- il set di istruzioni da assegnare ad ogni thread può risultare troppo oneroso in termini di computazioni e/o accessi in memoria
- studiare il cifrario può permettere di ottimizzare il calcolo perché performi bene in GPU

Attacco a PGP



Ottimizzazioni:

- porting su GPU per tutte le fasi dell'attacco
- identificazione nel sorgente di un controllo preliminare che filtra il 90% delle chiavi

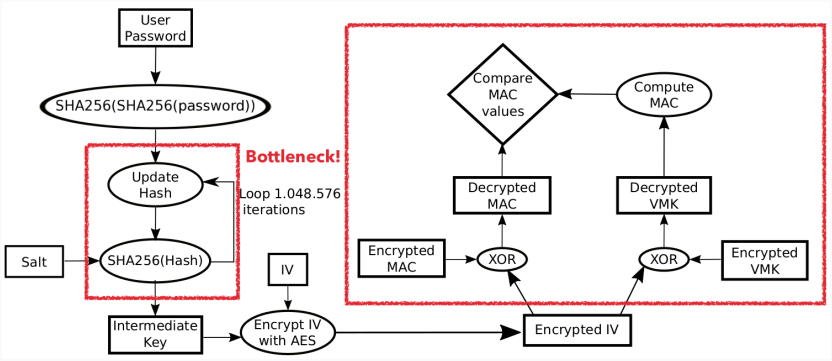


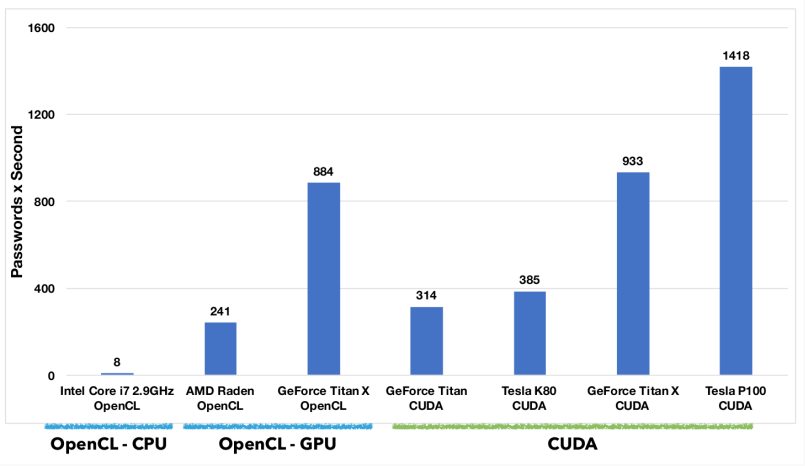
Risultati:

- generazione delle password: 6.400.000 password/sec.
- verifica della password: 400.000 password/sec. ($\times$1000 speed-up)

F. Milo, M. Bernaschi, M. Bisson: "A fast, GPU based, dictionary attack to OpenPGP secret keyrings"

BitCracker: Attacco a BitLocker





Ottimizzazioni:

- SHA-256 per GPU: migliore utilizzo memoria e riscrittura operazioni (no cicli e 32-bit integers)
- SHA-256 in BitLocker: precalcolo parte comune a tutte le chiamate a SHA-256 per stessa password
- trade-off precisione-efficienza: check primi 12 bytes invece di MAC test

Risultati su Tesla K80: da 80 a 385 password/sec. ($\sim 5\times$)

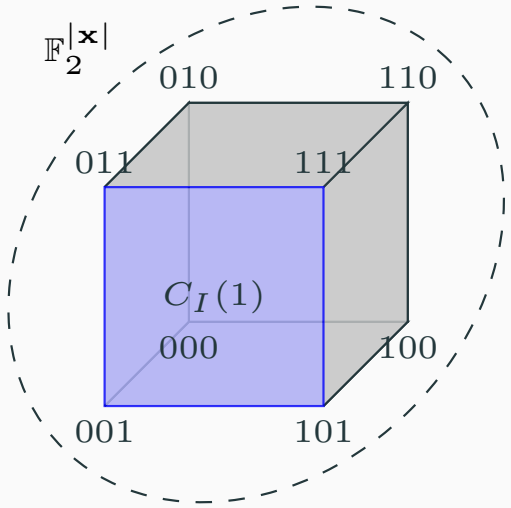
E. Agostini, M. Bernaschi: "BitCracker: BitLocker meets GPUs"

Ottimizzazione di Schemi di Attacco Generici: il Cube Attack

Il Cube Attack

Toy Example:

- $E(\mathbf{x}, \mathbf{y}) = p(\mathbf{x}, \mathbf{y}) = x_1x_2y_1y_2 + x_1x_2x_3y_1y_2 + x_1x_2y_1$
- $\mathbf{x} = (x_1, x_2, x_3)$ publiche, $\mathbf{y} = (y_1, y_2)$ private
- $\mathbf{x}_I = \{x_1, x_2\}$, fisso $\mathbf{x}_{\overline{I}} = x_3 = 1$: defisco il cubo $C_I(1)$
- $\sum_{\mathbf{v} \in C_I(1)} E(\mathbf{v}, \mathbf{y})$ al variare di $\mathbf{y}$: coefficienti del superpoly $p_{S(I)}(1, y_1, y_2) = (p(\mathbf{x}, \mathbf{y}) - q(\mathbf{x}, \mathbf{y}))/x_1x_2$
- se $p_{S(I)}(1, y_1, y_2)$ è lineare, x_1x_2 è un maxterm
- chosen-plaintext: ottengo $E(\mathbf{v}, K_1, K_2)$ per tutti i $\mathbf{v} \in C_I(1)$, calcolo $\Sigma_{K_{1,2}} = \sum_{\mathbf{v} \in C_I(1)} E(\mathbf{v}, K_1, K_2)$



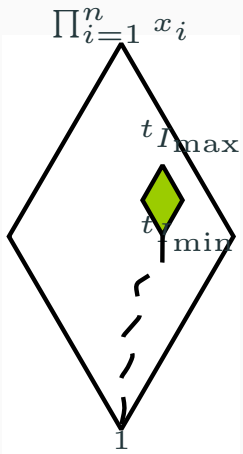
Ad alto livello:

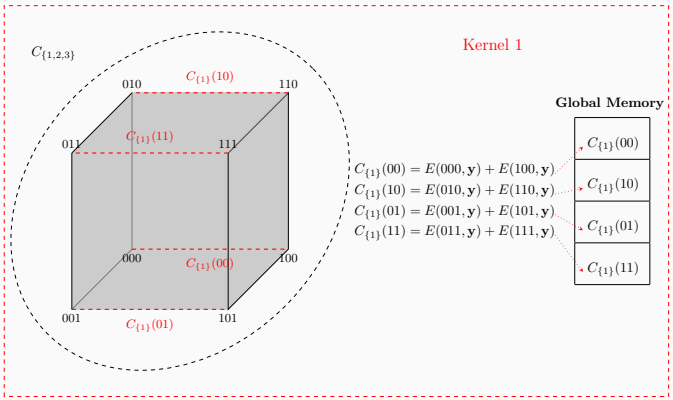
1. Cercare tanti maxterm (offline)
2. Per ogni maxterm, trovare i corrispondenti superpolinomi (offline)
3. Risolvere il sistema lineare ottenuto (online)

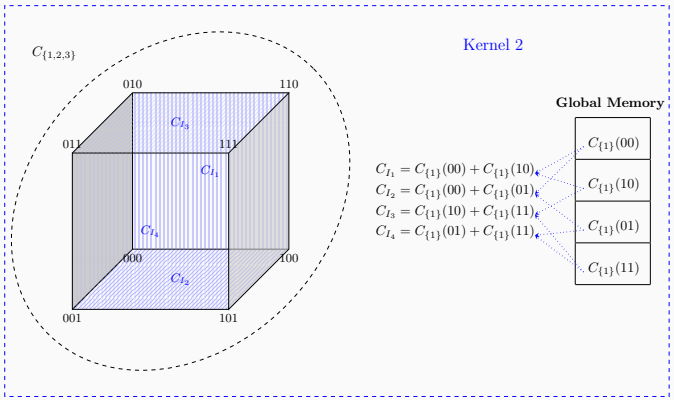
Kite-Attack: Cube Attack su GPU

Idea, problemi, soluzioni:

- SIMD: ogni thread calcola un cubo diverso
- calcolare un cubo richiede accesso a troppi dati
- condividere dati e/o risultati parziali tra thread è costoso
- esploriamo lo spazio tra $C_{I_{\min}}$ e $C_{I_{\max}}$
- 2 kernel distinti per scrittura e lettura di $C_{I_{\min}}(\mathbf{y})$
- inoltre: ottimizzazione del calcolo di $E(\mathbf{x}, \mathbf{y})$)



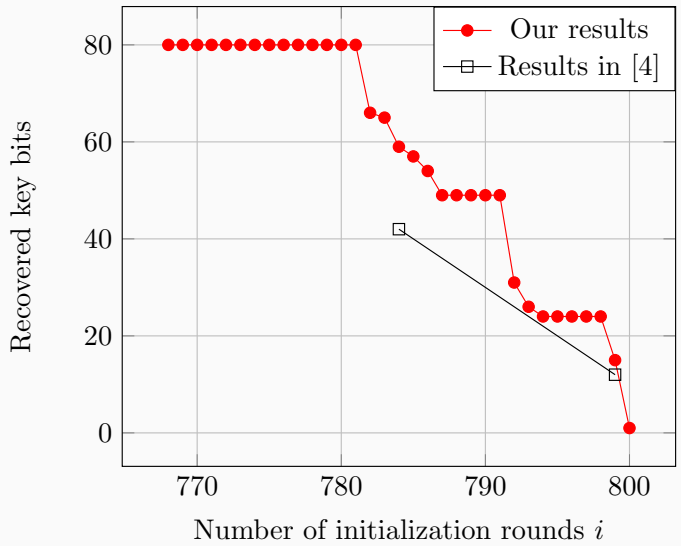




Kite Attack: Risultati

Risultati:

- migliorato lo stato dell'arte contro Trivium: full key recovery fino a 781 round, primo maxterm per Trivium-800
- Dimostrata portabilità su GPU, flessibilità su altri cifrari, vantaggio di una ricerca esaustiva sulle variabili pubbliche



M. Cianfriglia, S. Guarino, M. Bernaschi, F. Lombardi, M. Pedicini: "Kite attack: reshaping the cube attack for a flexible GPU-based maxterm search"

Conclusioni

Conclusioni

- ➕ La crittanalisi può giovare molto dell'utilizzo di calcolo ad alte prestazioni e parallelo
- ➕ Gli attacchi mostrati sono tutti scalabili su multi-GPU
- ➕ Le performance migliorano sensibilmente con l'utilizzo di architetture più moderne e recenti: lecito attendersi ulteriori miglioramenti futuri?
- ❓ La crittografia è pronta a difendersi?

Thanks! Any question?

Thank you for your attention!