Crittografia al Dipartimento di Matematica e Fisica di Roma Tre

Marco Pedicini (Roma Tre University)
partecipanti della sede:
Louis Nantenaina Andrianaivo (PhD student), Massimo Bernaschi (DR, IAC-CNR), Marco Cianfriglia (PhD student), Stefano Guarino (Assegnista IAC-CNR), Manoj Gyawali (PhD student), Flavio Lombardi (RIC, IAC-CNR), Francesca Merola (MAT/03), Francesco Pappalardi (MAT/02), Valerio Talamanca (MAT/02), Francesca Tartarone (MAT/02).

Meeting Conoscitivo
dell'Associazione De Componendis Cifris

22 January 2018, Rome, Italy

Outline

- Ricerca:
  - Crittoanalisi:
    - Attacchi algebrici: il Cube Attack di Shamir su $\mathbb{F}_2$;
    - Generalizzazione a $\mathbb{F}_q$;
    - Implementazione GPU in CUDA;
    - Applicazione per attaccare alcuni sistemi: Trivium, Grain128, Klein64.
  - Crittografia e Machine Learning:
    - il sistema di cifratura di Abadi (implementazione in TensorFlow);
    - un nuovo sistema ispirato dalla crittografia a chiave pubblica e allo schema BB84 (Bennet, Brassard 1984).
- Didattica:
  - Corsi: due corsi di laurea magistrale nella classe di laurea LM-40
    - Matematica,
    - Scienze Computazionali.

    entrambi prevedono un piano di studi nell'ambito della crittografia.
  - Tesi Laurea,
  - Dottorato
- Attività Internazionali.

Crittoanalisi

Attacchi Algebrici

Nella crittoanalisi si cerca un metodo per determinare la chiave utilizzata per cifrare (ripetutamente) con un certo **algoritmo**

$$E(x, k) = y.$$

Un metodo consiste nell'esprimere l'equazione che mette in relazione l'input (**plaintext** x e **chiave** k) con il cifrato y nella forma di un **polinomio su un campo** (finito).

Il successo del metodo dipende da vari fattori:

- dalla scelta del campo, e in base a questa scelta
- dalle dimensioni della rappresentazione (vettoriale) dell'input e dell'output dell'algoritmo.

A seconda delle dimensioni in gioco sarà necessario determinare più polinomi, i quali avranno un grado che determinerà il costo computazionale di risoluzione.

Generazione di Sistemi

Supponiamo di aver fissato un campo $\mathbb{F}$, e quindi di aver determinato le dimensioni di input ed output come vettori di $\mathbb{F}$.

Esempio: il caso tipico è $\mathbb{F} = \mathbb{F}_2$ e le dimensioni considerate sono le lunghezze in numero di bit delle rappresentazioni binarie di x, k e y.

Per ogni elemento y_j dell'output y resta fissato un polinomio a coefficienti in $\mathbb{F}$ che rappresenta la relazione

$$p_j(x, k) = y_j.$$

Con a disposizione una quantità sufficiente di coppie $(x_s, E(x_s, k^*))$, ovvero di **coppie plaintext e ciphertext** corrispondente cifrato con la **stessa chiave** k^*, sarà possibile impostare un **sistema algebrico** di equazioni :

$$p_j(x_s, k) = E(x_s, k^*)_j \text{ al variare di } x_s \text{ e } j$$

la cui soluzione (se determinato) permetterà di calcolare il vettore delle incognite k:

$$\begin{cases} p_1(x_1, k) & = E(x_1, k^*)_1 \\ & \vdots \\ p_m(x_n, k) & = E(x_n, k^*)_m \end{cases} \implies k = k^*.$$

Soluzione del Sistema

Affinchè sia tale, un attacco algebrico deve arrivare alla soluzione con un **costo computazionale inferiore** a quello dell'attacco a **forza bruta**.
Bisogna tenere conto

- sia del costo della **costruzione** della rappresentazione algebrica dell'algoritmo E (che però si può fare una volta per tutte indipendententemente dal traffico) e viene detta **fase offline**
- sia quello della **soluzione** del sistema in relazione al traffico osservato **fase online**.

Poiché il tipo di polinomi p_j trovati (in particolare il loro grado), influenza la **complessità della soluzione del sistema** (che è **polinomiale solo nel caso lineare**), sono stati cercati metodi per arrivare a sistemi equivalenti a quello dato ma con caratteristiche tali che ne semplifichino la soluzione (sistemi lineari).

Cube Attack

Nel 2008, Adi Shamir (in collaborazione con Itai Dinur) ha presentato un metodo per organizzare la fase offline in modo da ottenere a partire dal mero **accesso alla funzione di cifratura** E (attacco **black-box**), la costruzione di un sistema di equazioni lineari la cui soluzione fornisce la chiave.
Il campo di riferimento utilizzato è $\mathbb{F}_2$, si assume che l'**input** x è **controllabile**/selezionabile, e che esistano monomi m nelle variabili x per cui il polinomio $p(x,k)$ ammetta un **quoziente lineare** nelle incognite k

$$p_j(x,k) = mq_m(x,k) + r_m(x,k)$$

L'osservazione fondamentale per la costruzione è che **sommando** per **tutte** le **assegnazioni alle variabili del monomio** $\boxed{m = x_{i_1} \cdot \ldots \cdot x_{i_d}}$ si ottiene un'**equazione lineare**:

$$\sum_{(x_{i_1},\ldots,x_{i_d})=(0,\ldots,0)}^{(1,\ldots,1)} p_j(x,k) = q_m(x,k)$$

La ricerca dei monomi m

Il monomio m è determinato dagli indici $\boxed{I_m := \{i_1, \ldots, i_d\}}$ delle variabili, ed il quoziente $q_m(x, k)$ in realtà non dipende da tutte le variabili dell'input x ma solo da quelle che non appaiono nel monomio (ovvero, dalle rimanenti variabili di input controllabili x_i con $i \notin I_m$) e dalle variabili incognite k.

Un'osservazione importante è che le assegnazioni alle rimanenti variabili di x influenzano il **quoziente**, ed è dunque possibile che per alcune assegnazioni il quoziente sia **lineare** nelle incognite, mentre per altre sia **non lineare** oppure **costante**.

Per ogni scelta degli indici I_m, e per ogni assegnazione $x_i = b_i$ delle variabili con indice $i \notin I_m$ resta individuato un polinomio quoziente $q_{m,b}(k)$ nelle incognite k che potrebbe essere lineare, non lineare o costante.

Indichiamo con $x \otimes b$ il vettore delle variabili x con l'assegnazione delle variabili di indice i non in I_m a b_i e con $m \otimes b$ la scelta contemporanea del monomio m e dell'assegnazione delle rimanenti variabili a b.

Test di linearità e coefficienti (black box)

Per ogni scelta del monomio m ed ogni assegnazione b, bisogna **determinare** il polinomio $q_{m,b}(k)$ utilizzando **solo l'algoritmo** $E(x,k)$ e verificare che sia lineare. Infatti, il **cube attack** consiste nel **valutare** le **somme** sul **cubo** delle possibili **assegnazioni** alle variabili di indici in I_m, fissata un'assegnazione b delle rimanenti variabili e per opportune assegnazioni delle variabili private k:

- per $k = (0, \ldots, 0)$, ponendo tutti i bit di chiave a zero:

$$\sum_{(x_{i_1},\ldots,x_{i_d})=(0,\ldots,0)}^{(1,\ldots,1)} p_j(b,0) = q_{m,b}(0)$$

  che coincide con il termine α_0 di grado zero di $q_{m,b}(k)$;
- ponendo $k = \underline{i}$, l'i-esimo versore otteniamo

$$\sum_{(x_{i_1},\ldots,x_{i_d})=(0,\ldots,0)}^{(1,\ldots,1)} p_j(b,0) = q_{m,b}(\underline{i})$$

  che coincide con $\alpha_i + \alpha_0$ dove α_i è il coefficiente della variabile k_i

Test di linearità e coefficienti (black box)

Per ogni scelta del monomio m ed ogni assegnazione b, bisogna **determinare** il polinomio $q_{m,b}(k)$ utilizzando **solo l'algoritmo** $E(x,k)$ e verificare che sia lineare. Infatti, il **cube attack** consiste nel **valutare** le **somme** sul **cubo** delle possibili **assegnazioni** alle variabili di indici in I_m, fissata un'assegnazione b delle rimanenti variabili e per opportune assegnazioni delle variabili private k:

- per $k = (0, \ldots, 0)$, ponendo tutti i bit di chiave a zero:

$$\sum_{(x_{i_1},\ldots,x_{i_d})=(0,\ldots,0)}^{(1,\ldots,1)} E(x \otimes b, 0)_j = \sum_{(x_{i_1},\ldots,x_{i_d})=(0,\ldots,0)}^{(1,\ldots,1)} p_j(b,0) = q_{m,b}(0)$$

  che coincide con il termine α_0 di grado zero di $q_{m,b}(k)$;

- ponendo $k = \underline{i}$, l'i-esimo versore otteniamo

$$\sum_{(x_{i_1},\ldots,x_{i_d})=(0,\ldots,0)}^{(1,\ldots,1)} E(x \otimes b, \underline{i})_j = \sum_{(x_{i_1},\ldots,x_{i_d})=(0,\ldots,0)}^{(1,\ldots,1)} p_j(b,0) = q_{m,b}(\underline{i})$$

  che coincide con $\alpha_i + \alpha_0$ dove α_i è il coefficiente della variabile k_i

Linearità

Un test probabilistico per sapere se $q_{m,b}(k)$ è lineare in k consiste nel controllare per un certo numero di coppie di chiavi k_1 e k_2 l'uguaglianza:

$$\sum_{(x_{i_1},\ldots,x_{i_d})=(0,\ldots,0)}^{(1,\ldots,1)} E(x \otimes b, k_1 + k_2)_j + \alpha_0 =$$

$$= \sum_{(x_{i_1},\ldots,x_{i_d})=(0,\ldots,0)}^{(1,\ldots,1)} E(x \otimes b, k_1)_j +$$

$$+ \sum_{(x_{i_1},\ldots,x_{i_d})=(0,\ldots,0)}^{(1,\ldots,1)} E(x \otimes b, k_2)_j$$

Questo incrementa ulteriormente il numero di volte che la valutazione del cifrario deve essere effettuata al fine di calcolare le somme, tutto questo per valutare una singola scelta $m \otimes b$.
In totale, bisogna ripetere la valutazione per ogni vertice del cubo di dimensione d, per $|k| + 1$ volte per avere i coefficienti lineari e per un certo numero di chiavi (diciamo K) per effettuare il test di linearità:

$$2^d(K + |k| + 1) \text{ valutazioni dell'algoritmo di cifratura } E.$$

Riutilizzo delle valutazioni

- Se modifico la scelta del monomio e dell'assegnazione

$$m \otimes b \Longrightarrow m x_{i_{d+1}} \otimes b'$$

dove b' è la restrizione di b alle variabili diverse da $x_{i_{d+1}}$

- le nuove somme da calcolare sono legate alle precedenti:

$$\sum_{(x_{i_1},\ldots,x_{i_d},x_{i_{d+1}})=(0,\ldots,0,0)}^{(1,\ldots,1,1)} E(x \otimes b', k)_j =$$

$$= \sum_{(x_{i_1},\ldots,x_{i_d})=(0,\ldots,0)}^{(1,\ldots,1)} E(x \otimes b, k)_j + \sum_{(x_{i_1},\ldots,x_{i_d})=(0,\ldots,0)}^{(1,\ldots,1)} E(x \otimes \overline{b}, k)_j$$

dove $\overline{b}$ è l'assegnazione b complementata per il bit della variabile di indice i_{d+1}.

- Se salvo in memoria le somme precedenti legate alle scelte $m \otimes b$ e $m \otimes \overline{b}$ posso calcolare al costo di una somma aggiuntiva ripetuta per $K + |k| + 1$ volte quello che serve per valutare $m x_{i_{d+1}} \otimes b'$.

Kite Attack

In Cianfriglia M., Guarino, S., Bernaschi, M., Lombardi, F. and Pedicini, M. ***A Novel GPU-Based Implementation of the Cube Attack***, ACNS 2017, il cube attack è organizzato affinché sia efficacemente implementato su GPU.

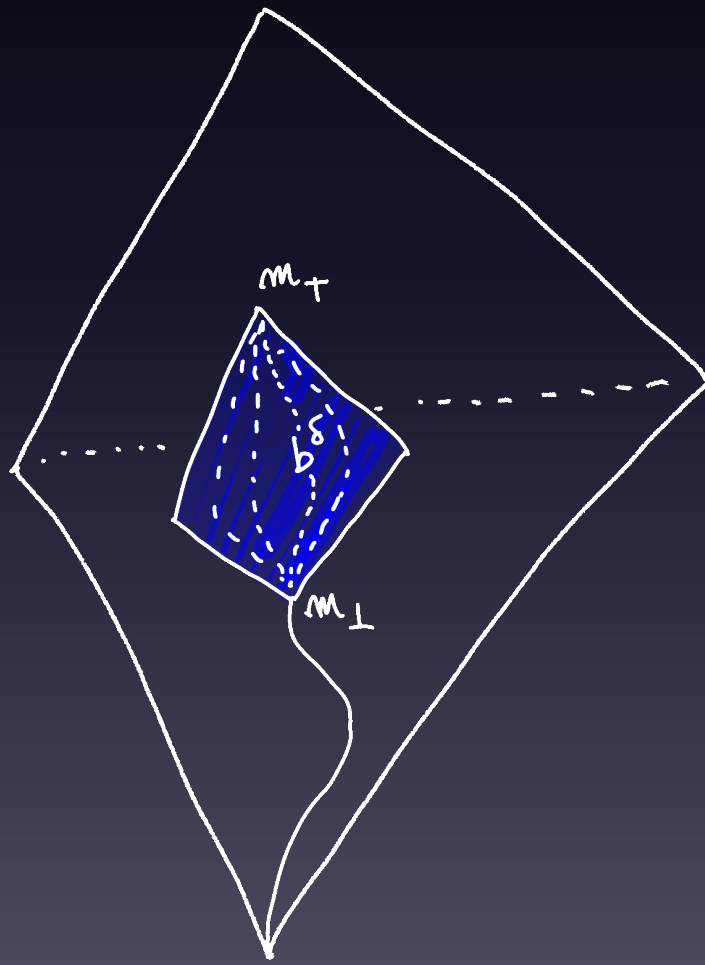
Il **Kite Attack** effettua una sorta di **Time-Memory Trade-Off** tra l'uso della memoria e il riutilizzo dei valori precalcolati:

- Si fissano $m_\perp$ e $m_\top$ in modo tale che $I_{m_\perp} \subset I_{m_\top}$.
- Si fissa anche un'unica assegnazione b_0 per la scelta $m_\top \otimes b_0$.
- Si pre-calcola e si salvano in memoria le valutazioni dei cubi per tutte le scelte $m_\perp \otimes (b \otimes b_0)$ al variare di tutte le assegnazioni b.
- Si ricombinano le somme parziali $m_\perp \otimes (b \otimes b_0)$ per ottenere le valutazioni per tutte le scelte intermedie $m \otimes (b^{(\delta)} \otimes b_0)$ con

$$I_{m_\perp} \subset I_m \subset I_{m^\top}$$

per ogni insieme di indici $\delta \subset I_{m_\top} \setminus I_{m_\perp}$.

Le dimensioni $a = |I_{m_\top}| - |I_{m_\perp}|$ e $b = |I_{m_\perp}|$ forniscono le informazioni sulla complessità in tempo e memoria dell'attacco, dunque talvolta diremo (a, b) **kite-attack**.



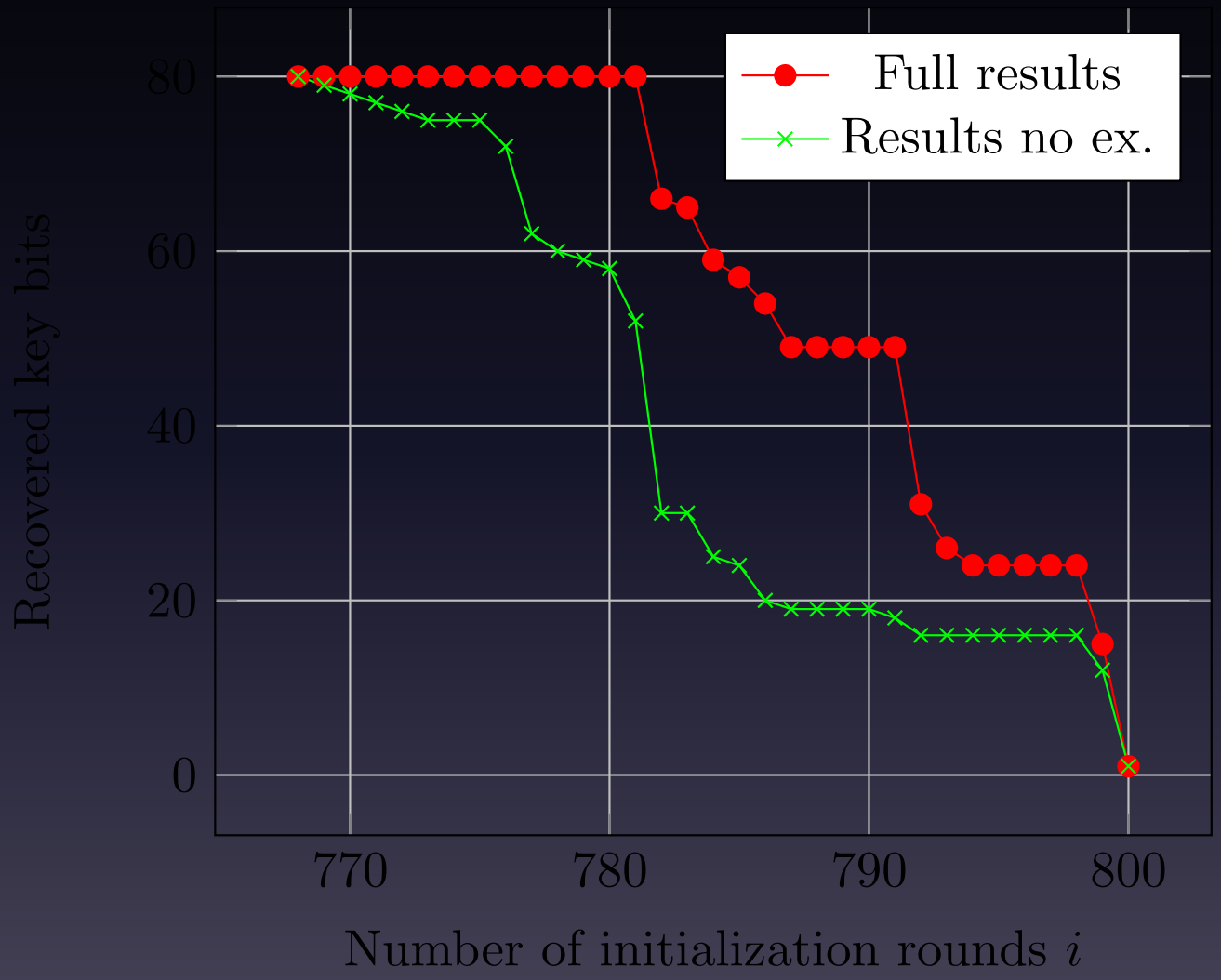
La memoria e la GPU

- Il calcolo con la GPU è organizzato:
  - in **warp**: l'unità di calcolo elementare della GPU formata da 32 thread;
  - ogni **thread** lavora su parole a 32-bit;
  - la **performance ottimale** si ottiene se tutti i 32 thread appartenenti alla stessa warp possono eseguire la **stessa istruzione** allo **stesso momento** su **dati diversi** ma che si trovano in aree di **memoria contigue**.
- La memoria di una scheda GPU:
  - una scheda ha **memoria globale limitata**, tra i 4 e i 24 GB;
  - l'**accesso casuale** alla memoria è supportato dalla scheda ma l'accesso è più **costoso**;
  - d'altra parte se i thread di uno stesso warp accedono **parole a 32bit consecutive** in memoria allora il **costo** per il thread è **unitario**;
  - inoltre l'accesso concorrente di più warp alle stesse locazioni di memoria (sia in lettura che in scrittura) comporta dei cicli di sincronizzazione e perciò rallenta il calcolo (**da evitare**).

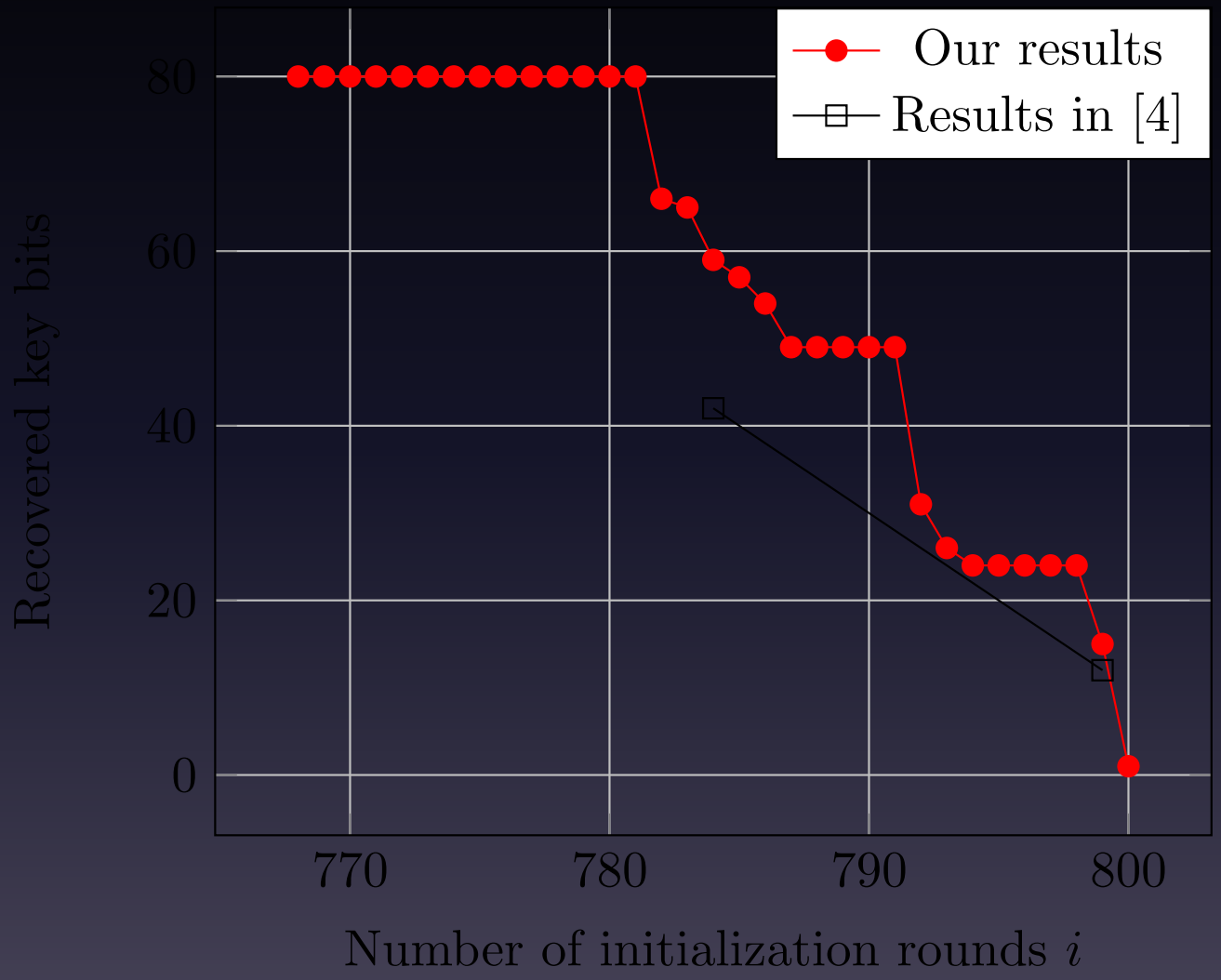
Applicazione a Trivium

- Trivium è uno strem cipher, disegnato da Bart Preenel, che genera il key stream 32 bit alla volta, per questo consideriamo $j \in \{1, \ldots, 32\}$ (nessun costo addizionale come tempo di esecuzione rispetto ad attaccare un polinomio alla volta);
- Analizziamo i 32 bit di key stream successivi ai 768 e 800 round di inizializzazione;
- Abbiamo effettuato un $(41, 25)$ kite-attack con 12 scelte di $m_\top$ e $m_\perp$ e $b_0 = 0$;
- La nostra implementazione utilizza 2 kernel GPU:
  - il primo kernel satura la memoria della GPU con il precalcolo delle scelte $m_\perp \otimes (b \otimes b_0)$ al variare di b in tutti i modi possibili,
  - il secondo kernel ricombina i valori calcolati in memoria per ottenere le scelte $m \otimes (b^{(\delta)} \otimes b_0)$
- Ogni run ha impiegato 55 ore di GPU per completare il calcolo di tutte le 2^{41} valutazioni.
- Grazie alla ricerca esaustiva per tutte le scelte intermedie $b^{(\delta)}$ abbiamo ottenuto un gran numero di polinomi lineari, e perciò un sistema di rango superiore rispetto alle versioni note del cube-attack (che tendono a fissare i bit fuori dal monomio a zero).

Risultati


Full results
Results no ex.
Recovered key bits
0
20
40
60
80
770
780
790
800
Number of initialization rounds i

Risultati


Our results
Results in [4]
Recovered key bits
0
20
40
60
80
770
780
790
800
Number of initialization rounds i

Kite attack per altri sistemi

Abbiamo iniziato ad analizzare :

- Grain-128 (increased size of the key and IV)
- Klein64 (block cipher)

Queste variazioni nelle caratteristiche dei crittosistemi considerati hanno un impatto sulla effettiva realizzazione su GPU del kite-attack.
D'altra parte anche le variazioni nelle caratteristiche delle schede GPU (in termini di memoria, numero di thread e set di istruzioni) possono influenzare le performance dell'implementazione.

Higher order Fields

In Agnesse, A. and Pedicini, M. **Cube attack in finite fields of higher order** (2011) è stato studiato il cube attack per campi finiti diversi da $\mathbb{F}_2$, l'implementazione su GPU dell'aritmetica in un generico $\mathbb{F}_q$ potrebbe essere più complicata, tuttavia l'attacco potrebbe ottenere risultati migliori se i polinomi che rappresentano l'algoritmo fossero più semplici per opportuni campi.

Didattica

LM in Matematica

Nella **Laurea Magistrale in Matematica** è previsto un percorso in crittografia e sicurezza delle informazioni

LAUREA MAGISTRALE I & II ANNO (120 CFU) Piano di Studi in Crittografia e Sicurezza delle informazioni
• 3 insegnamenti da 7 CFU a scelta (*) tra i seguenti (non sostenuti nel percorso della LT): {CR410, IN450, AC310, AL310, AL410, AL420, AM310, GE310, GE410, TN410}
• 1 insegnamento da 7 CFU a scelta (*) tra i seguenti (non sostenuti nel percorso della LT): {CR420, IN520, AN410, CP410, FM310}
• 2 ulteriori insegnamenti da 7 CFU *a scelta ampia* tra i seguenti (non sostenuti nel percorso della LT): {ALxxx, CRxxx, INxxx, TNxxx, LMxxx, ROxxx, ANxxx, CPxxx}
• 3 ulteriori insegnamenti da 7 CFU tra i seguenti (non sostenuti nel percorso della LT): {CRxxx, INxxx, CPxxx, ROxxx} di cui almeno 2 insegnamenti nel settore INxxx
• 1 insegnamento a scelta ampia da almeno 6 CFU anche all'esterno del corso di laurea
• QLMa/b (= Qualificazione alla Laurea Magistrale, ad idoneità) da 3 o 4 CFU
• UCL (=Ulteriori Conoscenze Linguistiche) da 5 CFU • IN530 da 4 CFU o Tirocinio da 4 CFU oppure • Tirocinio da 5 CFU • IN530 da 4 CFU oppure • Tirocinio da 7 CFU • ACUIMD (Altre conoscenze utili per l'inserimento nel mondo del lavoro) da 2 CFU
• PROVA FINALE 38 CFU

LM in Scienze Computazionali

Dall'a.a. 2017-18 abbiamo una **nuova Laurea Magistrale in Scienze Computazionali**.
Il Corso di Laurea in Scienze Computazionali è un corso Magistrale della Classe Matematica (LM 40); esso ha come obiettivo la formazione di un nuovo tipo di laureato con competenze interdisciplinari nei vari settori dell'informatica, della matematica applicata e della fisica.

Gli studenti avranno l'opportunità di studiare i fondamenti dell'informatica e del calcolo scientifico e di utilizzare le tecniche computazionali in un ampio spettro di aree applicative.

L'alto livello di specializzazione raggiunto al termine degli studi permette sia l'ingresso nel mondo del lavoro con competenze di tipo manageriale che l'accesso ai dottorati di ricerca italiani ed esteri con ottima qualificazione.

LM in Scienze Computazionali

L'elenco dei corsi da inserire in un **percorso** orientato alla crittografia nella laurea magistrale in **Scienze Computazionali**, prevede i seguenti esami:

- Gruppo B1) IN410, CR410, CR510;
- Gruppo B2) CP410, AN410;
- Gruppo C) IN450, IN480, IN490, IN520, IN420;
- Gruppo D) un esame a scelta dello studente.

Descrizione dei corsi

- Gruppo B1)
  - **IN410 - Modelli di Calcolo** (MAT/01, 7, 72 (60;12), c)
    Il corso di Modelli di Calcolo è dedicato all'approfondimento degli aspetti matematici del concetto di computazione, allo studio delle relazioni tra diversi modelli di calcolo e alla complessità computazionale.
  - **CR410 - Crittografia 1** (MAT/03, 7, 60, c)
    Acquisire una conoscenza di base dei concetti e metodi relativi alla teoria della crittografia a chiave pubblica, fornendo una panoramica di quelli che sono i modelli attualmente più utilizzati in questo settore
  - **CR510 - Crittosistemi ellittici** (MAT/02, 7, 60, c)
    Acquisire una conoscenza di base dei concetti e metodi relativi alla teoria della crittografia a chiave pubblica utilizzando il gruppo dei punti di una curva ellittica su un campo finito. Applicazioni della teoria delle curve ellittiche a problemi classici di teoria computazionale dei numeri come la fattorizzazione e i test di primalità.

Gruppo B2

- **CP410 - Probabilità 2** (MAT/06, 7, 60, b/c)
  Acquisire una solida preparazione negli aspetti principali della teoria della probabilità: costruzione di misure di probabilità su spazi misurabili, legge 0-1, indipendenza, aspettazioni condizionate, variabili casuali, convergenza di variabili casuali, funzioni caratteristiche, teorema del limite centrale, processi di ramificazione e alcuni risultati fondamentali nella teoria delle martingale a tempo discreto.
- **AN410 - Analisi numerica 1** (MAT/08, 7, 72 (60; 12), b/c)
  Dare gli elementi fondamentali (inclusa la implementazione in un linguaggio di programmazione) delle tecniche di approssimazione numerica di base, in particolare quelle legate alla soluzione di sistemi lineari e di equazioni scalari non lineari, all'interpolazione e alla integrazione approssimata.

Gruppo C

- **IN490 - Linguaggi di Programmazione** (INF/01, 7, 60, c)
  Presentare i principali concetti della teoria dei linguaggi formali e la loro applicazione alla classificazione dei linguaggi di programmazione. Introdurre le principali tecniche per l'analisi sintattica dei linguaggi di programmazione. Imparare a riconoscere la struttura di un linguaggio di programmazione e le tecniche per implementarne la macchina astratta. Conoscere il paradigma orientato agli oggetti ed un altro paradigma non imperativo.
- **IN480 - Calcolo Parallelo e Distribuito** (INF/01, 7, 60, (48,12), c)
  Acqusire le tecniche di programmazione parallela e distribuita, e la conoscenza delle moderne architetture hardware e software per il calcolo scientifico ad alte prestazioni. Introdurre i metodi iterativi distribuiti per la simulazione di problemi numerici. Acquisire la conoscenza dei linguaggi di nuova concezione per la programmazione dinamica nel calcolo scientifico, quali il linguaggio Julia.
- **IN450 - Algoritmi per la crittografia** (INF/01, 7, 60, c)
  Acquisire la conoscenza dei principali algoritmi di cifratura. Approfondire le competenze matematiche necessarie alla descrizione degli algoritmi. Acquisire le tecniche di crittoanalisi utilizzate nella valutazione del livello di sicurezza fornito dai sistemi di cifratura.
- **IN520 - Tecniche di sicurezza dei dati e delle reti** (ING-INF/03, 7, 60, c)
  Introdurre i concetti fondamentali della sicurezza e la capacità di poter autonomamente aggiornare le proprie conoscenze nel dominio sicurezza dei dati e delle reti. Fornire i concetti di base per la comprensione e la valutazione di soluzione di sicurezza. Fornire le conoscenze per poter produrre soluzioni di sicurezza per sistemi di piccole/medie dimensioni.

Gruppo C (ulteriori corsi)

- **IN420 - Teoria dell'Informazione** (INF/01, 7, 60, c)
  Introdurre questioni fondamentali della teoria della trasmissione dei segnali e nella loro analisi quantitativa. Concetto di entropia e di mutua informazione. Mostrare la struttura algebrica soggiacente. Applicare i concetti fondamentali alla teoria dei codici, alla compressione dei dati e alla crittografia
- **GE460 - Teoria dei grafi** (MAT/03, 7, 60, b/c)
  Fornire strumenti e metodi della teoria dei grafi
- **IN430 - Tecniche informatiche avanzate** (INF/01, 7, 60, c)
  Acquisire le capacità concettuali di strutturare un problema secondo il paradigma ad oggetti. Acquisire la capacità di produrre il disegno di soluzioni algoritmiche basate sul paradigma ad oggetti. Acquisire i concetti di base relativi a tecniche di programmazione basate sul paradigma ad oggetti. Introdurre i concetti fondamentali di programmazione parallela e concorrente.
- **IN440 - Ottimizzazione Combinatoria** (INF/01, 7, 60, c)
  Acquisire competenze sulle principali tecniche di risoluzione per problemi di ottimizzazione combinatoria; approfondire le competenze sulla teoria dei grafi; acquisire competenze tecniche avanzate per la progettazione, l'analisi e l'implementazione al calcolatore di algoritmi per la risoluzione di problemi di ottimizzazione su grafi, alberi e reti di flusso.

Gruppo D) un esame a scelta.

Tesi di Laurea

1. Edoardo Fasano, ***Neural Networks and cryptography***, Master 2 in Mathematics, Department of Mathematics and Physics, Roma Tre University (defended on October 2017).
2. Gioia Pierdomenico, ***Multi-party Computation and applications to authentication infrastructures***, Master 2 in Mathematics, Department of Mathematics and Physics, Roma Tre University (defended on October 2017).
3. Giulia Maragnani, ***Reticoli, Crittosistemi e l'Algoritmo LLL in Crittoanalisi***, Master 2 in Mathematics, Department of Mathematics and Physics, Roma Tre University (defended on January 2017).
4. Raffaella Cuomo, ***Crittoanalisi Lineare e Differenziale e Resistenza di Funzioni Pseudorandom***, Master 2 in Mathematics, Department of Mathematics and Physics, Roma Tre University (defended on January 2016).
5. Dario Giannini, ***Parallel Implementation of RSA Challenges***, Master 2 in Mathematics, Department of Mathematics and Physics, Roma Tre University (defended on February 2015).
6. Stefania Todisco, ***Advances in automatic cryptanalysis of block ciphers***, Master 2 in Mathematics, Department of Mathematics and Physics, Roma Tre University (defended on October 2013).
7. Marco Vargiu, ***Fast Algebraic Cryptanalysis in Finite Fields of Higher Order with the Cube Attack***, Master 2 in Mathematics, Department of Mathematics and Physics, Roma Tre University (defended on February 2013).
8. Stefano Guarino, ***Ciphertext-only reconstruction of LFSR-based stream ciphers***, Master 2 in Mathematics, Dept. of Mathematics, Faculty of Sciences, Roma Tre University (defended on February 2010).
9. Simona Giovannetti, ***Cryptographic hash functions: algorithms for collision searching and lambda-calculus optimal reduction***, Master 2 in Mathematics, Dept. of Mathematics, Faculty of Sciences, Roma Tre University (defended on May 2009).
10. Andrea Agnesse, ***Stream Ciphers: from Correlation Attacks to the Cube Attack***, Master 2 in Mathematics, Dept. of Mathematics, Faculty of Sciences, Roma Tre University (defended on May 2009).
11. Rosina Otranto, ***Cifrari a flusso e loro crittoanalisi***, Laurea quadriennale in Mathematics, Dept. of Mathematics, Faculty of Sciences, Roma Tre University (defended on July 2006).
12. Mario Mento, ***Advanced Encryption Standard: punti di forza e debolezze***, Laurea quadriennale in Mathematics, Dept. of Mathematics, Faculty of Sciences, Roma Tre University (defended on October 2005).