SAPIENZA
UNIVERSITÀ DI ROMA

FACOLTÀ DI INGEGNERIA DELL'INFORMAZIONE,
INFORMATICA E STATISTICA

Sviluppo di un sistema per la decrittazione del traffico telefonico GSM

Relatore:
Prof. Massimo Bernaschi

Candidato:
Flavio Pietrelli

Sviluppo di un sistema per la decrittazione del traffico telefonico GSM

SAPIENZA
UNIVERSITÀ DI ROMA
FACOLTÀ DI INGEGNERIA DELL'INFORMAZIONE, INFORMATICA E STATISTICA

Sviluppo di un sistema per la decrittazione del traffico telefonico GSM

Relatore:
Prof. Massimo Bernaschi

Candidato:
Flavio Pietrelli

Buongiorno, mi chiamo Flavio Pietrelli, sono uno studente del corso di laurea Magistrale e mi presento con la tesi dal titolo "Sviluppo di un sistema per la decrittazione del traffico telefonico GSM".

Obiettivo:

Sviluppare un sistema per la decrittazione del traffico telefonico GSM e l'estrazione dell'audio della conversazione.

Organizzazione della presentazione:

- Illustrazione del sistema crittografico del GSM basato sull'algoritmo di cifratura A5/1.
- Accenno alle principali tecniche di attacco al cifrario A5/1 e descrizione delle "Berlin A5/1 rainbow table set" e della suite Kraken.
- Presentazione del software GSMCrack e panoramica sulle fasi in cui è organizzato l'attacco.
- Conclusioni e sviluppi futuri.

Obiettivo:

Sviluppare un sistema per la decrittazione del traffico telefonico GSM e l'estrazione dell'audio della conversazione.

Organizzazione della presentazione:

- Illustrazione del sistema crittografico del GSM basato sull'algoritmo di cifratura A5/1.
- Accenno alle principali tecniche di attacco al cifrario A5/1 e descrizione delle "Berlin A5/1 rainbow table set" e della suite Kraken.
- Presentazione del software GSMCrack e panoramica sulle fasi in cui è organizzato l'attacco.
- Conclusioni e sviluppi futuri.

L'obiettivo di questa tesi è per l'appunto quello di sviluppare un sistema che permetta la decrittazione del traffico telefonico GSM e quindi l'estrazione e l'ascolto dell'audio della conversazione.

La presentazione è suddivisa in 4 sezioni organizzate in una prima parte di illustrazione del sistema crittografico A5/1 adottato nel GSM.

Una seconda parte in cui si fa accenno alle principali tecniche di attacco al cifrario A5/1 e alle cosiddette "Berlin tables" utilizzate in questa tesi.

Una terza parte in cui è presentato GSMCrack, il software sviluppato per automatizzare le diverse fasi in cui è organizzato l'attacco.

Ed infine alcune considerazioni sui risultati ottenuti ed i possibili sviluppi futuri di questo lavoro.

Il sistema GSM

- Nato nel 1982.
- È lo standard di telefonia mobile più diffuso al mondo:
  - Più di 200 paesi.
  - Oltre 4 miliardi di utenti.
- Segna l'inizio della comunicazione mobile digitale.
- Introduce il concetto di sicurezza in termini di:

Riservatezza dell'identità del cliente	Autenticazione	Confidenzialità dei dati trasmessi sul canale radio

Il sistema telefonico GSM nasce nel 1982 come standard di 2ª generazione e, ad oggi, rappresenta lo standard di telefonia mobile più diffuso al mondo.

Il suo sviluppo ha segnato l'inizio della comunicazione mobile digitale e per la prima volta ha introdotto il concetto di sicurezza in termini di:

Riservatezza dell'identità del cliente

Autenticazione alla rete

e confidenzialità dei dati trasmessi sul canale di comunicazione.

E' proprio su quest'ultimo aspetto che si è sviluppato il lavoro di tesi.

Decrittazione del traffico telefonico GSM

È un processo che può essere scomposto in quattro fasi:

- Intercettazione e raccolta delle informazioni trasmesse sul canale radio.
- Ricerca di coppie «*plaintext*, *ciphertext*» per eseguire un attacco di tipo *known-plaintext* al cifrario A5/1.
- Esecuzione dell'attacco e recupero della chiave di sessione utilizzata per la cifratura dei dati.
- Decrittazione dei messaggi scambiati sul canale di comunicazione ed estrazione dell'audio della conversazione.

Decrittazione del traffico telefonico GSM

È un processo che può essere scomposto in quattro fasi:

- Intercettazione e raccolta delle informazioni trasmesse sul canale radio.
- Ricerca di coppie «*plaintext*, *ciphertext*» per eseguire un attacco di tipo *known-plaintext* al cifrario A5/1.
- Esecuzione dell'attacco e recupero della chiave di sessione utilizzata per la cifratura dei dati.
- Decrittazione dei messaggi scambiati sul canale di comunicazione ed estrazione dell'audio della conversazione.

La decrittazione del traffico telefonico GSM è un processo che può essere scomposto in quattro fasi:

L'intercettazione e la raccolta delle informazioni trasmesse sul canale radio.

La ricerca di informazioni utili ad eseguire un attacco di tipo known-plaintext ai danni del cifrario A5/1.

Si esegue poi l'attacco vero e proprio recuperando la chiave di sessione utilizzata per la cifratura dei dati.

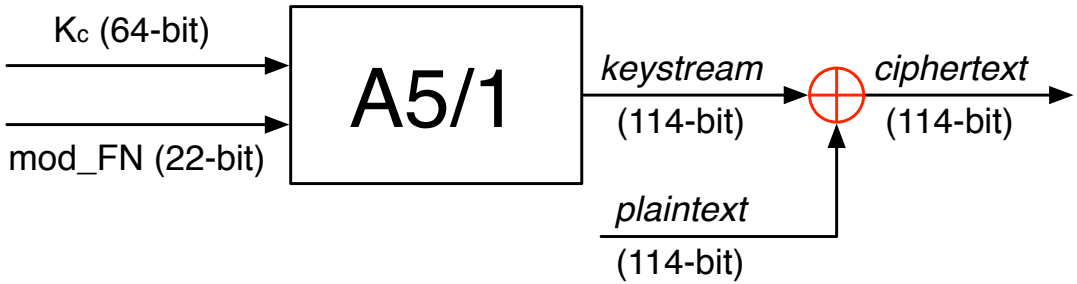
E, infine, si utilizza la chiave trovata per decrittare i messaggi cifrati ed estrarre quindi l'audio della conversazione.

Durante questa tesi ci si è concentrati prevalentemente sugli ultimi tre punti e quindi sull'esecuzione vera e propria dell'attacco.

Vediamo ora come è implementata la crittografia all'interno del sistema GSM e come eseguire l'attacco al cifrario utilizzato.

Il cifrario A5/1

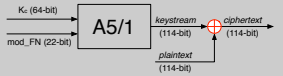
L'algoritmo A5/1 si occupa di cifrare/decifrare i dati trasmessi sul canale radio.



- È un algoritmo a chiave simmetrica di tipo *stream cipher*.
- Riceve in input:
  - La chiave di sessione K_c.
  - Il valore *mod_FN* associato al *frame number*.
- Produce la chiave di cifratura *keystream* (114 bit).
- Esegue la cifratura come XOR tra il *plaintext* e la *keystream*.
- Per ogni *burst* è utilizzata una *keystream* differente.

Il cifrario A5/1

L'algoritmo A5/1 si occupa di cifrare/decifrare i dati trasmessi sul canale radio.



- È un algoritmo a chiave simmetrica di tipo *stream cipher*.
- Riceve in input:
  - La chiave di sessione K_c.
  - Il valore *mod_FN* associato al *frame number*.
- Produce la chiave di cifratura *keystream* (114 bit).
- Esegue la cifratura come XOR tra il *plaintext* e la *keystream*.
- Per ogni *burst* è utilizzata una *keystream* differente.

La cifratura dei dati trasmessi sul canale radio è affidata ad un algoritmo che prende il nome di Algoritmo A5/1 ed è qui rappresentato schematicamente.

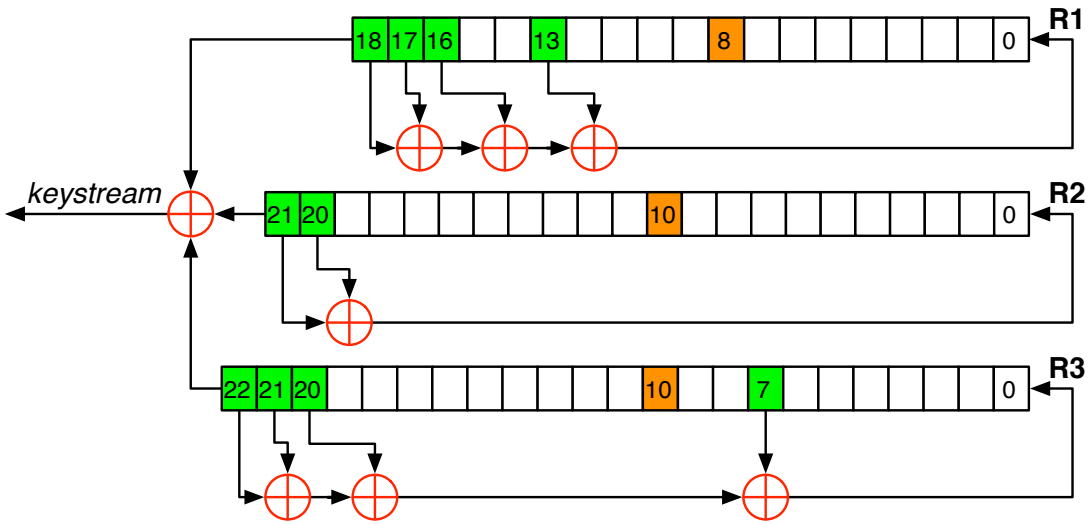
Si tratta di un algoritmo di cifratura a chiave simmetrica di tipo stream cipher che riceve in input due stringhe binarie: una chiave di sessione K_c ed un valore numerico derivato dal frame number del messaggio considerato. In output produce una chiave di cifratura chiamata keystream di lunghezza 114 bit.

La cifratura è eseguita come XOR tra la keystream e 114 bit del plaintext.

E' importante sottolineare che durante la comunicazione ogni messaggio viene diviso in sequenze di 114 bit e ognuna viene cifrata utilizzando una keystream (e quindi una chiave di cifratura) differente l'una dall'altra.

Linear Feedback Shift Registers

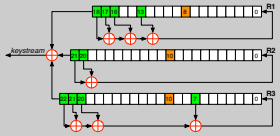
La *keystream* è ottenuta grazie all'uso combinato di 3 LFSR:



- I registri vengono inizializzati inserendo in sequenza:
  - La chiave di sessione K_c.
  - Il valore *mod_FN* associato al *frame number*.
- Lo scorrimento è irregolare e gestito mediante una funzione di maggioranza: $maj(a, b, c) = (a \times b) \oplus (b \times c) \oplus (c \times a)$

Linear Feedback Shift Registers

La *keystream* è ottenuta grazie all'uso combinato di 3 LFSR:



- I registri vengono inizializzati inserendo in sequenza:
  - La chiave di sessione K_c.
  - Il valore *mod_FN* associato al *frame number*.
- Lo scorrimento è irregolare e gestito mediante una funzione di maggioranza: $maj(a, b, c) = (a \times b) \oplus (b \times c) \oplus (c \times a)$

Quello mostrato in questa figura rappresenta l'interno dell'algoritmo A5/1. La keystream è generata bit a bit per mezzo dell'uso combinato di 3 cosiddetti "linear feedback shift registers", tradotto in italiano come "registri a scorrimento con retroazione lineare", ognuno dei quali è sostanzialmente un generatore pseudocasuale di bit.

La dimensione totale dei registri è di 64 bit e il loro utilizzo è caratterizzato una fase preliminare di inizializzazione, che prevede l'inserimento in sequenza della chiave di sessione K_c ed il valore associato al frame number e, successivamente, da uno scorrimento irregolare, gestito mediante una funzione di maggioranza, che, ad ogni passaggio, causa l'avanzamento di almeno due dei tre registri.

Queste accortezze permettono all'algoritmo di creare di volta in volta una chiave di cifratura articolata e difficilmente invertibile.

Attacco al cifrario A5/1

Lo scopo è quello di ottenere lo stato interno dei registri LFSR che ha generato la *keystream* e, da questo, recuperare la chiave di sessione K_c utilizzata per la cifratura dei dati.

Diversi tipi di attacco sono possibili:

- Brute-Force (complessità di 2^{64}):
  - spazio su disco non richiesto
  - nessuna pre-computazione iniziale
  - risultato garantito
  - tempo di esecuzione molto lungo
- Code-Book ($\sim$128 PB):
  - velocità di attacco
  - risultato garantito
  - pre-computazione iniziale onerosa
  - enorme quantità di memoria
- Time-Memory trade-off:
  - velocità di attacco
  - pre-computazione iniziale
  - discreta quantità di memoria

Attacco al cifrario A5/1

Lo scopo è quello di ottenere lo stato interno dei registri LFSR che ha generato la *keystream* e, da questo, recuperare la chiave di sessione K_c utilizzata per la cifratura dei dati.

Diversi tipi di attacco sono possibili:

- Brute-Force (complessità di 2^{64}):
  - spazio su disco non richiesto
  - nessuna pre-computazione iniziale
  - risultato garantito
  - tempo di esecuzione molto lungo
- Code-Book (∼128 PB):
  - velocità di attacco
  - risultato garantito
  - pre-computazione iniziale onerosa
  - enorme quantità di memoria
- Time-Memory trade-off:
  - velocità di attacco
  - pre-computazione iniziale
  - discreta quantità di memoria

In questa diapositiva sono mostrati pregi e difetti delle principali tecniche di attacco utilizzabili; per tutti, lo scopo è quello di ottenere lo stato interno dei registri che ha generato una specifica keystream e, da questo, recuperare la chiave di sessione K_c inserita durante la fase di inizializzazione e utilizzata per generare tutte le keystream.

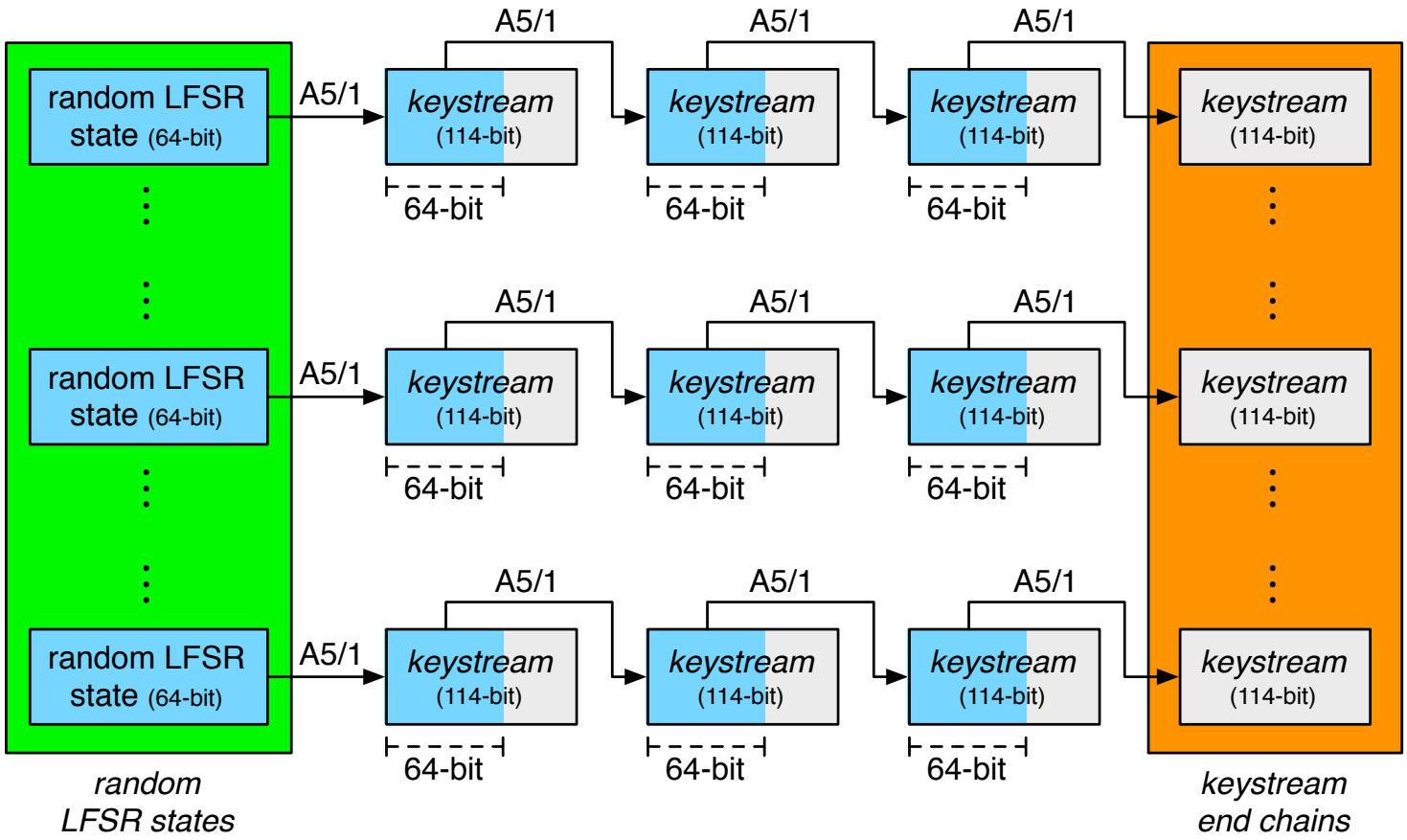
Un'evoluzione del classico attacco "a forza bruta" è rappresentato dal cosiddetto "code book", ovvero, una sorta di dizionario che ad ogni stato interno dei registri associa la chiave di cifratura da essi prodotta. Un dizionario di questo tipo una volta realizzato permetterebbe la ricerca in pochi secondi anche su un comune PC. La sua realizzazione comporta però una fase di pre-elaborazione non indifferente e circa 128 PB di spazio disco, rendendolo di fatto impraticabile e meno pratico persino di un attacco a forza bruta.

Un'ottimizzazione dell'approccio "code book" consiste, invece, nel trovare un compromesso tra "tempo computazionale" speso durante la ricerca e "spazio di memoria" utilizzato su disco. Questo in crittografia viene chiamato time-memory trade-off (o appunto "compromesso tempo-memoria").

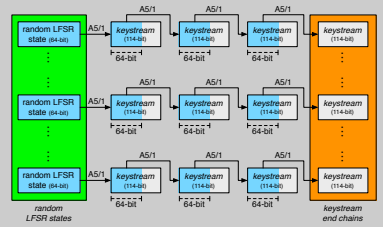
L'idea è quella di non mantenere su disco l'intero dizionario con tutti gli stati associati a tutte le keystream, bensì di conservare un dizionario contenente solo una porzione dei dati e demandando il calcolo dei rimanenti alla fase di ricerca della chiave.

Questa tecnica si pone quindi a metà tra un attacco a forza bruta, in cui i dati vengono ricalcolati ogni volta, ed un attacco di tipo "code-book", in cui i dati vengono calcolati un'unica volta e poi riutilizzati per le successive esecuzioni dell'attacco.

Time-Memory trade-off



- Catene lunghe comportano:
  - minore quantità di memoria
  - tempo di ricerca più lungo
- Catene corte comportano:
  - tempo di ricerca più breve
  - maggiore quantità di memoria



In questa immagine è rappresentata in maniera schematica la struttura interna di un dizionario basato sul concetto del Time Memory trade-off.

Si tratta di un dizionario organizzato in lunghe catene, ognuna delle quali inizia con uno stato dei registri scelto in maniera casuale.

Allo stato di inizio catena viene quindi applicato l'algoritmo A5/1 e, della keystream appena prodotta, si considerano i primi 64 bit come un nuovo stato dei registri sul quale eseguire nuovamente l'algoritmo A5/1. Ripetendo il procedimento più volte si genera l'intera catena, fino a raggiungere una lunghezza prestabilita.

Il dizionario ottenuto viene quindi compresso conservando solo la prima e l'ultima colonna, demandando poi il calcolo di quelle centrali alla fase di ricerca della chiave.

Con un dizionario di questo tipo è quindi possibile trovare un giusto compromesso tra la quantità di dati da conservare su disco ed il numero di chiavi da ricalcolare durante la fase di ricerca.

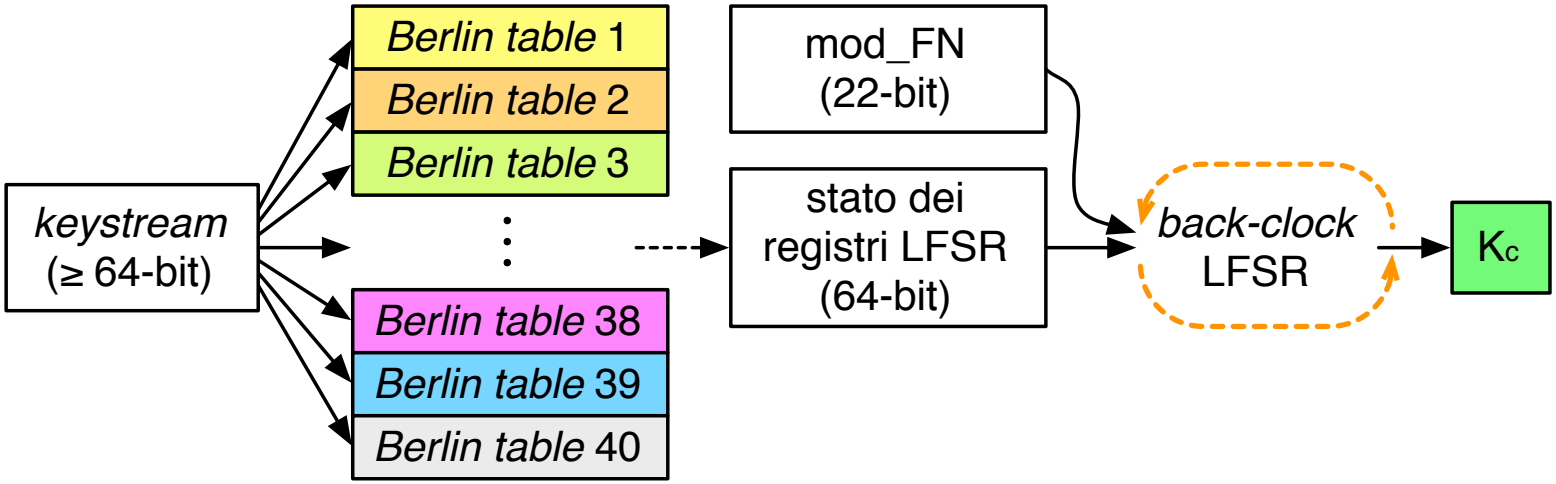
Catene più lunghe, infatti, necessitano di minor spazio su disco, ma causano tempi di ricerca più lunghi. Viceversa catene più corte permettono tempi di ricerca più brevi, ma necessitano di maggior spazio occupato su disco.

Berlin A5/1 rainbow table set

Costituiscono il cuore dell'attacco al cifrario A5/1.

L'idea è quella di creare un insieme di tabelle che associa ("mappa") ogni possibile sequenza di 64 bit che compone lo stato interno dei registri LFSR ai primi 64 bit della *keystream* da essi prodotta.

Conoscendo il *frame number* e lo stato interno dei registri è possibile effettuare il *back-clock* dei tre LFSR e risalire alla chiave di sessione K_c utilizzata.

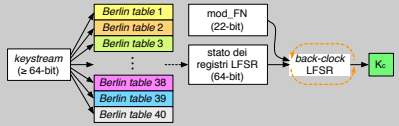


Berlin A5/1 rainbow table set

Costituiscono il cuore dell'attacco al cifrario A5/1.

L'idea è quella di creare un insieme di tabelle che associa ("mappa") ogni possibile sequenza di 64 bit che compone lo stato interno dei registri LFSR ai primi 64 bit della *keystream* da essi prodotta.

Conoscendo il *frame number* e lo stato interno dei registri è possibile effettuare il *back-clock* dei tre LFSR e risalire alla chiave di sessione K_c utilizzata.



Il dizionario utilizzato per questa tesi prende il nome di "Berlin A5/1 rainbow table set" o più semplicemente "Berlin tables" e consiste in un insieme di tabelle che sfruttano il concetto del time-memory trade-off per associare ogni possibile sequenza di 64 bit che compone lo stato interno dei registri ai primi 64 bit della keystream da essi prodotta.

Ottenuto lo stato interno dei registri e conoscendo il frame number del messaggio, che è un parametro noto, è quindi possibile eseguire quello che viene chiamato "back-clock" dei registri e risalire quindi alla chiave di sessione K_c inserita durante la fase di inizializzazione e utilizzata per generare tutte le keystream.

Berlin A5/1 rainbow table set

Si presentano al download come un insieme di 40 tabelle:

- Ognuna ha una dimensione di 42 GB.
- La dimensione totale è di circa 1.7 TB.
- La copertura stimata è pari a circa il 19% dello spazio totale degli stati, ovvero 2^{61} possibili chiavi.
  - Dimezzare il numero di chiavi implica raddoppiare il numero delle catene calcolate.
- Per essere utilizzate necessitano di un processo preliminare di indicizzazione.
- Ogni tabella è identificata da un ID univoco:

100	108	116	124	132	140	148	156	164	172
180	188	196	204	212	220	230	238	250	260
268	276	284	292	324	332	340	348	356	364
372	380	388	396	404	412	420	428	492	500

Berlin A5/1 rainbow table set

Si presentano al download come un insieme di 40 tabelle:

- Ognuna ha una dimensione di 42 GB.
- La dimensione totale è di circa 1.7 TB.
- La copertura stimata è pari a circa il 19% dello spazio totale degli stati, ovvero 2^{61} possibili chiavi.
  - Dimezzare il numero di chiavi implica raddoppiare il numero delle catene calcolate.
- Per essere utilizzate necessitano di un processo preliminare di indicizzazione.
- Ogni tabella è identificata da un ID univoco:

100	108	116	124	132	140	148	156	164	172
180	188	196	204	212	220	230	238	250	260
268	276	284	292	324	332	340	348	356	364
372	380	388	396	404	412	420	428	492	500

Le “Berlin tables”, nella versione attuale, si presentano al download come un insieme di 40 tabelle ognuna di dimensione 42 Gigabyte per un totale di circa 1.7 Terabyte. Nonostante le dimensioni particolarmente generose, la copertura stimata è pari a circa il 19% dello spazio totale degli stati, ovvero 2^{61} possibili chiavi.

Aumentare la copertura è possibile costruendo altre tabelle, ma con la tecnica attuale è stimato che per dimezzare il numero di chiavi assenti sarebbe necessario raddoppiare il numero delle catene calcolate, con importanti conseguenze sull’occupazione di memoria.

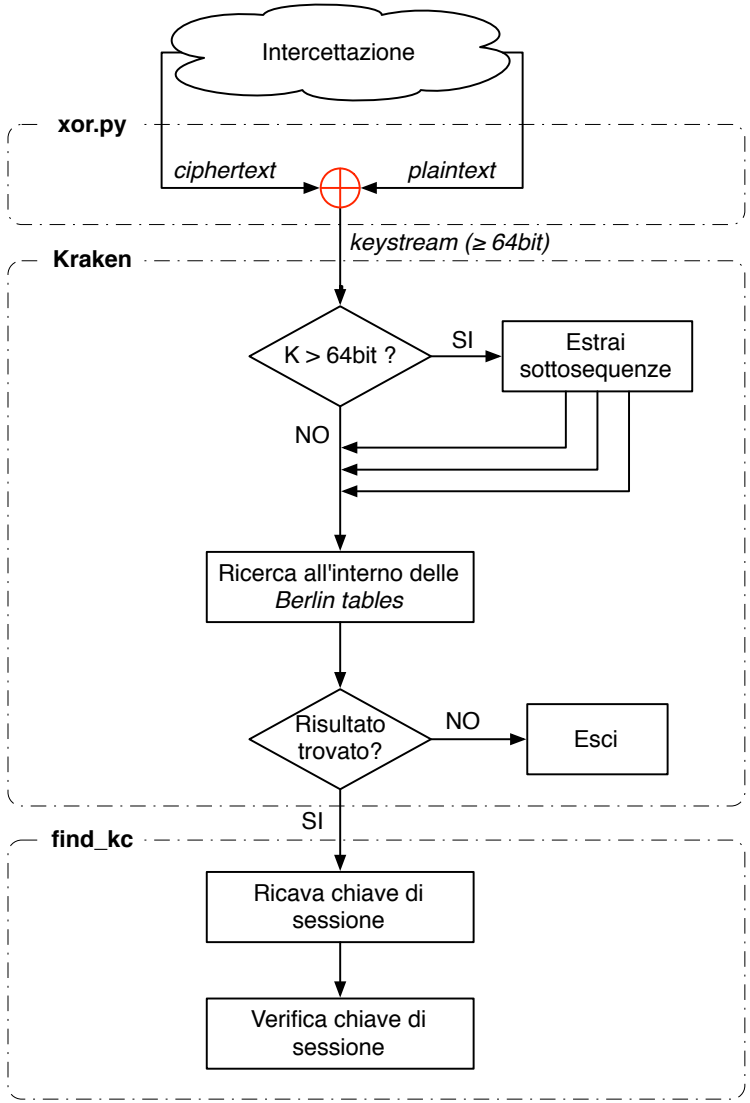
Vista la dimensione di queste tabelle, il loro utilizzo prevede una fase preliminare di indicizzazione e questo, assieme la ricerca all’interno di esse è possibile per mezzo di una suite di programmi dedicati che prende il nome di Kraken.

Kraken

Fornisce gli strumenti per utilizzare le *Berlin tables* ed eseguire l'attacco al cifrario A5/1.

Permette di:

- Indicizzare le *Berlin tables*.
- Eseguire l'operazione di XOR tra stringhe binarie.
- Eseguire la ricerca di una *keystream* all'interno delle *Berlin tables*.
- Eseguire l'operazione di *back-clock* dei registri e ricavare la chiave di sessione K_c.

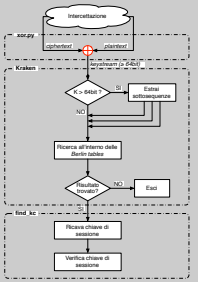


Kraken

Fornisce gli strumenti per utilizzare le *Berlin tables* ed eseguire l'attacco al cifrario A5/1.

Permette di:

- Indicizzare le *Berlin tables*.
- Eseguire l'operazione di XOR tra stringhe binarie.
- Eseguire la ricerca di una *keystream* all'interno delle *Berlin tables*.
- Eseguire l'operazione di *back-clock* dei registri e ricavare la chiave di sessione K_c.



Kraken è appunto una suite e fornisce tutti gli strumenti necessari per utilizzare le tabelle e completare l'attacco al cifrario A5/1. Grazie ad essa è infatti possibile indicizzare le tabelle per prepararle al loro utilizzo, eseguire la ricerca di una keystream all'interno di esse e, in caso di successo, eseguire anche l'operazione di back-clock dei registri e calcolare quindi la chiave di sessione K_c utilizzata per la cifratura.

Decrittazione del traffico telefonico GSM

Utilizzare le *Berlin tables* per eseguire la decrittazione del traffico telefonico GSM è un'operazione che comporta diverse difficoltà:

- Sono necessari dati in chiaro per eseguire un attacco di tipo *known-plaintext*: $P \oplus C = P \oplus (P \oplus K) = (P \oplus P) \oplus K = K$

Come fare?

- I messaggi *System Information Type 5/5ter/6* sono degli ottimi candidati:
  - Trasportano sempre le stesse informazioni.
  - Sono inviati ad intervalli regolari.
  - Sono inviati sia prima che dopo l'attivazione della crittografia.
- La registrazione GSM deve iniziare prima dell'attivazione della crittografia.

Decrittazione del traffico telefonico GSM

Utilizzare le *Berlin tables* per eseguire la decrittazione del traffico telefonico GSM è un'operazione che comporta diverse difficoltà:

- Sono necessari dati in chiaro per eseguire un attacco di tipo *known-plaintext*: $P \oplus C = P \oplus (P \oplus K) = (P \oplus P) \oplus K = K$

Come fare?

- I messaggi *System Information Type 5/5ter/6* sono degli ottimi candidati:
  - Trasportano sempre le stesse informazioni.
  - Sono inviati ad intervalli regolari.
  - Sono inviati sia prima che dopo l'attivazione della crittografia.
- La registrazione GSM deve iniziare prima dell'attivazione della crittografia.

Spiegato come funziona l'attacco al cifrario A5/1 è ora giunto il momento di applicarlo ad un contesto reale.

Utilizzare le *Berlin tables* per eseguire la decrittazione del traffico telefonico GSM è però un'operazione che comporta diverse difficoltà. Tra queste la sfida più grande è proprio ottenere la keystrem da utilizzare per la ricerca all'interno delle tabelle.

L'idea è quella di eseguire un attacco di tipo known-plaintext e quindi di raccogliere una serie di dati cifrati di cui si conosce il contenuto e con questi eseguire nuovamente l'operazione di XOR (questa volta tra plaintext e ciphertext) per risalire alla keystream corrispondente.

Ottenere questi dati non è un operazione impossibile e anzi il GSM offre diverse possibilità per ottenere dati cifrati il cui contenuto è noto o comunque facilmente ricostruibile. Tra queste la soluzione qui utilizzata è quella di sfruttare i messaggi di tipo "System information type 5/5ter/6" inviati dalla stazione radio al dispositivo mobile.

Questi sono dei messaggi di controllo che hanno la caratteristica di trasportare sempre le stesse informazioni, ma soprattutto sono inviati ad intervalli regolari sia prima che dopo l'attivazione della crittografia.

Ottenuti questi messaggi in chiaro, è quindi possibile andare a ricercare il messaggio corrispondente nella parte cifrata e ricavare quindi la keystream utilizzata. Per fare questo è ovviamente necessario che la registrazione GSM inizi prima dell'attivazione della crittografia.

GSMCrack

Permette di eseguire in maniera del tutto automatica l'attacco al cifrario A5/1 e la decrittazione "off-line" di traffico telefonico GSM precedentemente registrato.

Scritto in linguaggio C/C++ è utilizzabile su sistemi UNIX.

Si avvale di diversi software di supporto quali:

- Kraken
- AirProbe
- libosmocore
- GSMFrameCoder
- Toast

È soggetto ai limiti imposti da AirProbe:

- Decodifica del solo traffico in *downlink* (dalla stazione radio verso il dispositivo mobile).
- Non gestisce il *frequency hopping*.

GSMCrack

Permette di eseguire in maniera del tutto automatica l'attacco al cifrario A5/1 e la decrittazione "off-line" di traffico telefonico GSM precedentemente registrato.

Scritto in linguaggio C/C++ è utilizzabile su sistemi UNIX.

Si avvale di diversi software di supporto quali:

- Kraken
- AirProbe
- libosmocore
- GSMFrameCoder
- Toast

È soggetto ai limiti imposti da AirProbe:

- Decodifica del solo traffico in *downlink* (dalla stazione radio verso il dispositivo mobile).
- Non gestisce il *frequency hopping*.

GSMCrack rappresenta il lavoro conclusivo di questa tesi ed è l'elemento che conferisce originalità a questa ricerca. Si tratta, infatti, di un software sviluppato per eseguire in maniera del tutto automatica l'attacco al cifrario A5/1 e la decrittazione offline di traffico telefonico GSM fino all'estrazione dell'audio della conversazione.

E' un programma scritto in linguaggio C/C++ per sistemi UNIX e si avvale di diversi altri software di supporto di cui i più rilevanti sono:

Kraken, per la ricerca all'interno delle berlin tables ed il recupero della chiave di sessione K_c e Airprobe, utilizzato per la decodifica e l'analisi del file contenente la registrazione GSM.

L'utilizzo di AirProbe pone alcuni limiti all'esecuzione dell'attacco permettendo ad esempio la decodifica del solo il traffico in downlink (cioè quello dalla stazione radio vero il dispositivo mobile) e l'impossibilità di gestire il frequency hopping. Va comunque detto che si tratta di restrizioni di carattere puramente implementativo ed è quindi probabile che nuove funzionalità possano essere aggiunte con un aggiornamento futuro dell'applicazione.

Organizzazione dell'attacco

Fase 1:

- Ricerca e decodifica del canale di controllo utilizzato.
- Ricerca dei messaggi di *System Information Type 5/5ter/6*.

Fase 2:

- Ricerca di possibili coppie «*plaintext*, *ciphertext*».
- Aggiornamento del *Timing Advance*.
- Calcolo delle *keystream* ($K = P \oplus C$).
- Ricerca delle *keystream* all'interno delle *Berlin tables*.
- Estrazione della chiave di sessione K_c.

Fase 3:

- Ricerca, decodifica e decrittazione del canale di traffico utilizzato.
- Estrazione dell'audio della conversazione.

Organizzazione dell'attacco

Fase 1:
- Ricerca e decodifica del canale di controllo utilizzato.
- Ricerca dei messaggi di *System Information Type 5/5ter/6*.

Fase 2:
- Ricerca di possibili coppie «*plaintext*, *ciphertext*».
- Aggiornamento del *Timing Advance*.
- Calcolo delle *keystream* ($K = P \oplus C$).
- Ricerca delle *keystream* all'interno delle *Berlin tables*.
- Estrazione della chiave di sessione K_c.

Fase 3:
- Ricerca, decodifica e decrittazione del canale di traffico utilizzato.
- Estrazione dell'audio della conversazione.

In questa diapositiva sono riassunte le 3 fasi in cui con GSMCrack è stato organizzato l'attacco:

La prima fase consiste prevalentemente nella ricerca dei messaggi in chiaro di tipo "System information type 5/5ter/6".

Trovati uno o più di questi messaggi si procede all'esecuzione vera e propria dell'attacco e quindi alla ricerca dei corrispondenti messaggi cifrati per ottenere la keystream utilizzata. Per evitare una ricerca puramente esaustiva, GSMCrack sfrutta la caratteristica di questi messaggi, di essere inviati ad intervalli regolari e si considerano solo quelli che si trovano alla giusta distanza gli uni dagli altri.

Le keystream calcolate vengono quindi verificate con Kraken per la ricerca all'interno delle berlin tables.

Ottenuta la chiave di sessione, si procede poi alla ricerca e decrittazione del canale di traffico e quindi all'estrazione dell'audio della conversazione.

GSMCrack

Perché utilizzare GSMCrack?

È l'unico strumento che permette di eseguire la decrittazione del traffico telefonico GSM in maniera del tutto automatizzata.

Principali caratteristiche:

- Semplicità d'uso.
- Minima interazione da parte dell'utente.
- Gestione "intelligente" delle ricerche:
  - Ottimizzazione nella ricerca di coppie «*plaintext*, *ciphertext*».
  - Ottimizzazione nella gestione del *Timing Advance*.
  - Filtro sulle *keystream* già verificate.
- Interruzione e ripristino dell'esecuzione.
- Log delle operazioni.

GSMCrack

Perché utilizzare GSMCrack?

È l'unico strumento che permette di eseguire la decrittazione del traffico telefonico GSM in maniera del tutto automatizzata.

Principali caratteristiche:

- Semplicità d'uso.
- Minima interazione da parte dell'utente.
- Gestione "intelligente" delle ricerche:
  - Ottimizzazione nella ricerca di coppie «*plaintext*, *ciphertext*».
  - Ottimizzazione nella gestione del *Timing Advance*.
  - Filtro sulle *keystream* già verificate.
- Interruzione e ripristino dell'esecuzione.
- Log delle operazioni.

Tra le principali caratteristiche di GSMCrack vi sono sicuramente l'efficienza e la semplicità d'uso.

GSMCrack è, infatti, l'unico software che permette di eseguire la decrittazione del traffico telefonico GSM in maniera del tutto automatizzata e con la minima interazione da parte dell'utente.

Inoltre, le scelte implementative adottate per la gestione delle ricerche permettono di ottimizzare le prestazioni, massimizzando le possibilità di successo e al tempo stesso riducendo il più possibile il numero di confronti da effettuare e di keystream da verificare.

Tra le altre caratteristiche vi è poi un meccanismo di interruzione e ripristino dell'esecuzione.

Ed un file di log per tenere traccia di tutte le operazioni compiute dal programma.

Risultati sperimentali

1 hard disk di dimensione 3 TB:

	Ricerca esaustiva			GSMCrack			
	«P,C»	keys.	Tempo	«P,C»	keys.	Tempo	Risparmio
File 1	15	60	2h30m15s	3	12	30m28s	400%
File 2	269	1074	44h32m13s	19	74	3h05m26s	1351%
File 3	744	2974	123h14m46s	24	94	3h53m26s	3064%
File 4	12	45	1h50m05s	4	13	31m17s	246%

Tempo impiegato per la ricerca di una *keystream*: $\sim$150 sec.

7 hard disk di dimensione 300 GB:

	Ricerca esaustiva			GSMCrack			
	«P,C»	keys.	Tempo	«P,C»	keys.	Tempo	Risparmio
File 1	15	60	22m00s	3	12	4m24s	400%
File 2	269	1074	6h33m48s	19	74	27m08s	1351%
File 3	744	2974	18h10m28s	24	94	34m28s	3064%
File 4	12	45	16m30s	4	13	4m46s	246%

Tempo impiegato per la ricerca di una *keystream*: $\sim$22 sec.

1 hard disk di dimensione 3 TB:

	Ricerca esaustiva			GSMCrack			
	«P,C»	keys.	Tempo	«P,C»	keys.	Tempo	Risparmio
File 1	15	60	2h30m15s	3	12	30m28s	400%
File 2	269	1074	44h32m13s	19	74	3h05m26s	1351%
File 3	744	2974	123h14m46s	24	94	3h53m26s	3064%
File 4	12	45	1h50m05s	4	13	31m17s	246%

Tempo impiegato per la ricerca di una *keystream:* ∼150 sec.

7 hard disk di dimensione 300 GB:

	Ricerca esaustiva			GSMCrack			
	«P,C»	keys.	Tempo	«P,C»	keys.	Tempo	Risparmio
File 1	15	60	22m00s	3	12	4m24s	400%
File 2	269	1074	6h33m48s	19	74	27m08s	1351%
File 3	744	2974	18h10m28s	24	94	34m28s	3064%
File 4	12	45	16m30s	4	13	4m46s	246%

Tempo impiegato per la ricerca di una *keystream:* ∼22 sec.

L'efficienza di GSMCrack è stata testata utilizzando due differenti configurazioni hardware: La prima con le Berlin Tables contenute in un unico disco che consente a Kraken una ricerca pressoché lineare. La seconda suddividendo le tabelle su 7 dischi e permettendo quindi a Kraken di parallelizzare la ricerca su questi.

In queste tabelle sono mostrati e confrontati i risultati ottenuti dalla decrittazione di 4 registrazioni GSM eseguendo la stessa prova dapprima utilizzando una ricerca puramente esaustiva, e successivamente utilizzando GSMCrack.

Con 1 hard disk, considerando per esempio il file 3, nel caso della ricerca esaustiva vengono confrontate oltre 2900 chiavi impiegando circa 5 giorni di elaborazione continua. Con GSMCrack le chiavi confrontate sono solo 94 ed il tempo di esecuzione scende a poco meno di 4 ore. Quasi 31 volte di meno.

Aumentando il numero di hard disk il numero di confronti rimane ovviamente lo stesso, ma il tempo di esecuzione crolla drasticamente. Considerando sempre il file 3, con questa configurazione la ricerca esaustiva impiega circa 18 ore di esecuzione, mentre con GSMCrack appena 34 minuti.

E' importante sottolineare che il tempo di esecuzione dipende comunque dal tempo impiegato per la ricerca della singola keystream all'interno delle berlin tables. Utilizzando un hard disk la ricerca di una chiave impiega in media 2 minuti e mezzo, mentre con 7 hard disk si scende a circa 22 secondi.

L'ottimo sarebbe quello di utilizzare 40 hard disk allo stato solido il che permetterebbe presumibilmente di effettuare la ricerca di una chiave in poco meno di un secondo.

Conclusioni:

- Dimostrata l'effettiva vulnerabilità del sistema crittografico adottato nel GSM.
- Eseguita la decrittazione del traffico telefonico GSM arrivando ad estrarre l'audio della conversazione.
- Emersi i limiti imposti dall'utilizzo di AirProbe per l'acquisizione e la decodifica dei dati GSM.

Sviluppi futuri:

- Acquistare un dispositivo USRP1/2 per continuare la ricerca su un dataset più corposo di registrazioni.
- Estendere l'attacco alle più recenti reti di 3ª generazione.
- Sviluppare un sistema ottimizzato per la creazione di nuove tabelle.

Conclusioni:
- Dimostrata l'effettiva vulnerabilità del sistema crittografico adottato nel GSM.
- Eseguita la decrittazione del traffico telefonico GSM arrivando ad estrarre l'audio della conversazione.
- Emersi i limiti imposti dall'utilizzo di AirProbe per l'acquisizione e la decodifica dei dati GSM.

Sviluppi futuri:
- Acquistare un dispositivo USRP1/2 per continuare la ricerca su un dataset più corposo di registrazioni.
- Estendere l'attacco alle più recenti reti di 3ª generazione.
- Sviluppare un sistema ottimizzato per la creazione di nuove tabelle.

I risultati ottenuti sono nel complesso molto positivi e confermano l'effettiva vulnerabilità del sistema crittografico adottato nel GSM.

Durante il periodo di tesi non abbiamo avuto la possibilità di eseguire l'acquisizione dei dati dalla rete e per questo per lo sviluppo di GSMCrack e le prove condotte sono state utilizzate delle registrazioni effettuate da altri.

Continuando questo studio anche dopo il periodo di tesi, il prossimo passo è dunque l'acquisto di un dispositivo di acquisizione per continuare la ricerca su un insieme più corposo di registrazioni.

Ulteriori sviluppi futuri sono inoltre quelli di tentare di estendere l'attacco alle più recenti reti di 3ª generazione il cui algoritmo di cifratura è al momento violato solo accademicamente.

E, successivamente, di sviluppare anche un sistema di generazione delle tabelle che sia più efficiente di quello attuale il quale, per generare le Berlin Tables ha impiegato 8 GPU e più di un mese di elaborazione continua.

SAPIENZA
UNIVERSITÀ DI ROMA

FACOLTÀ DI INGEGNERIA DELL'INFORMAZIONE,
INFORMATICA E STATISTICA

Sviluppo di un sistema per la decrittazione del traffico telefonico GSM

Relatore:
Prof. Massimo Bernaschi

Candidato:
Flavio Pietrelli

Sviluppo di un sistema per la decrittazione del traffico telefonico GSM

SAPIENZA
UNIVERSITÀ DI ROMA
FACOLTÀ DI INGEGNERIA DELL'INFORMAZIONE, INFORMATICA E STATISTICA

Sviluppo di un sistema per la decrittazione del traffico telefonico GSM

Relatore:
Prof. Massimo Bernaschi

Candidato:
Flavio Pietrelli

Grazie :)