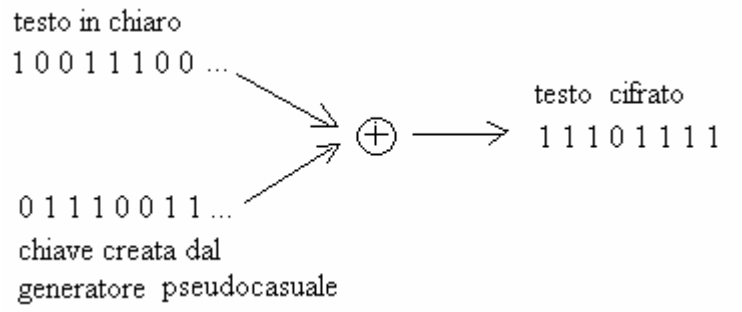
REGISTRI A SCORRIMENTO LINEARE

I cifrari perfetti, come per esempio, il one time pad, per essere sicuri è fondamentale che i bit, che rappresentano la chiave per cifrare (o decifrare) un messaggio, siano scelti casualmente quindi creati da un generatore di bits casuali.

Però anche se questo sistema appare perfetto, viene usato raramente in quanto presenta degli svantaggi:

- Troppo dispendioso dal punto di vista economico e di impiego delle risorse.
- La chiave segreta da trasmettere è molto lunga e visto che deve essere trasmessa segretamente, non va bene.

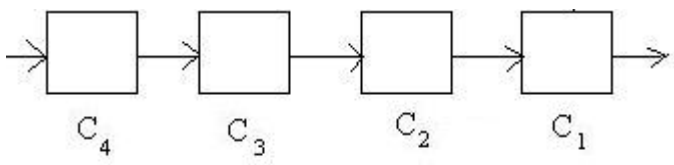
Allora al fine di risolvere questo ultimo problema si pensò di sostituire la sequenza completamente casuale con una che è solo pseudocasuale.
La cosa buona è che una sequenza di questo genere appare come se fosse casuale anche se in realtà non lo è, ma la cosa più importante è che essa è determinata da un numero molto piccolo di parametri: quindi il mittente deve trasmettere solo questi pochi parametri che forniscono la chiave.



Cioè si fa la somma bit a bit tra la sequenza che rappresenta il testo in chiaro e la sequenza che rappresenta la chiave generata casualmente.
Quello che ottengo è la cifratura del testo. Analogamente se conosco la sequenza di bit della chiave posso sommarla bit a bit con il testo cifrato per riavere il testo in chiaro.
(questo è analogo a ciò che succedeva per le sequenze casuali)
Con questa procedura il problema dello scambio della chiave, anche se ridotto, resta.
Un altro problema è che questi sistemi non sono più perfettamente sicuri come i precedenti.
Così si deve trovare un compromesso tra livello di sicurezza e la quantità di dati d trasmettere.
Quasi tutte le chiavi pseudocasuali usate in crittografia vengono generate da registri a scorrimento perché:

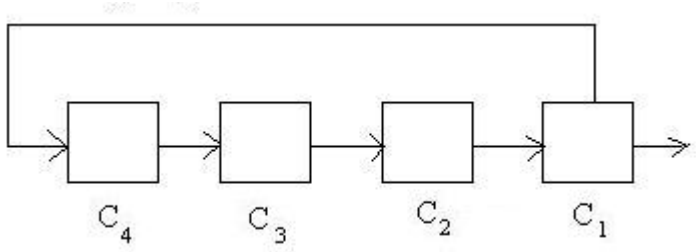
- I registri a scorrimento possono essere realizzati in maniera molto efficace via hardware
- La teoria matematica dei registri a scorrimento si è ben sviluppata e risulta essere abbastanza chiara.

Vediamo l'esempio più semplice di registro a scorrimento di lunghezza 4:

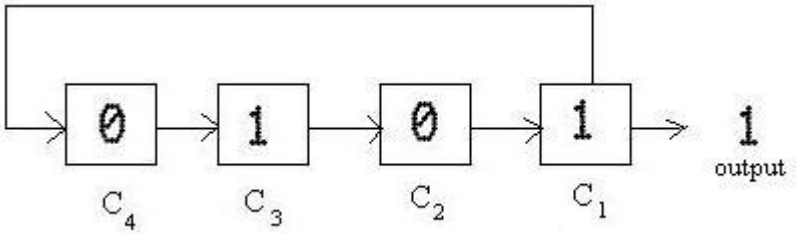


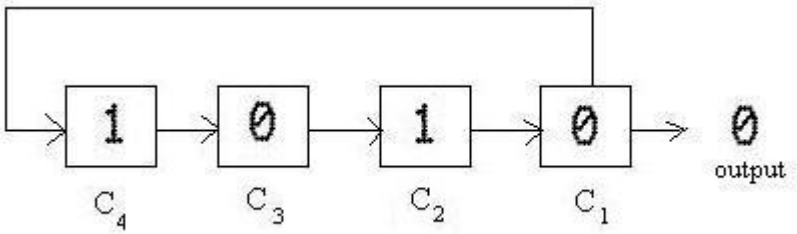
I registri a scorrimento sono controllati da un clock e ad tempo fissato il contenuto di ogni cella passa nella cella successiva. La cella più a destra è la prima, quella più a sinistra è l'ultima.

Capiamo subito che non si può usare una struttura così semplice poiché dopo 4 clock il registro risulterebbe vuoto.
Vediamo quest'altra struttura:

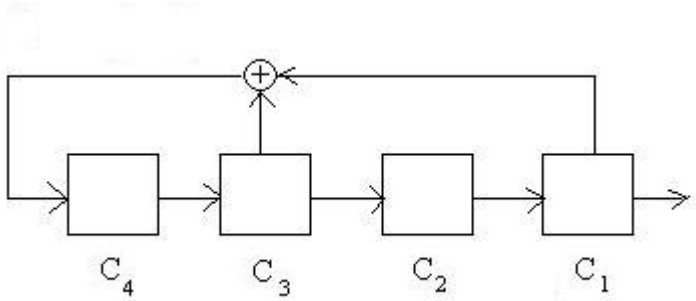


In cui il contenuto della prima cella oltre ad andare in output viene rinviato con una funzione di retroazione (feedback) all'ultima cella.
In questo modo però il registro darebbe periodicamente il suo stato iniziale.





Vediamone un'altra:

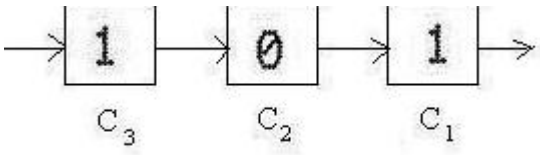


Qui l'ultima cella riceve in input non solo dalla prima ma anche dalla terza. Più precisamente l'input che riceve l'ultima cella è la somma dello stato corrente della prima cella e della terza.
Le celle che non danno contributo alla funzione di retroazione si occupano solamente di prendere al tempo i+1 il contenuto della cella precedente al tempo i.
Ora vediamo un esempio di come avvengono questi shift e quindi come si genera la sequenza che avremo in output, cioè la chiave generata.
Prendiamo come esempio un registro di lunghezza n=4 con stato iniziale: 0 0 0 1.

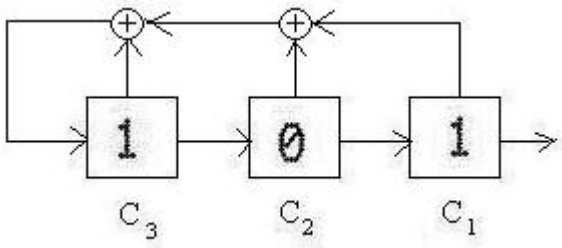
CRITTOANALISI DEI REGISTRI A SCORRIMENTO LINEARI

Facciamo un esempio di crittoanalisi nei registri a scorrimento lineari.
Pensiamo che un crittoanalista conosca la lunghezza di un registro a scorrimento da cui è stata ottenuta una chiave per cifrare il testo.
Per ricostruire la chiave deve conoscere lo stato iniziale del registro e la funzione di retroazione.
Consideriamo quindi la sequenza di 6 bit: 101001. e pensiamo di sapere che sia generata da un registro di lunghezza 3

- La prima osa da fare è individuare lo stato iniziale del registro.(cosa molto semplice perché i primi bits prodotti da un registro sono quelli che occupano le celle nello stato iniziale)



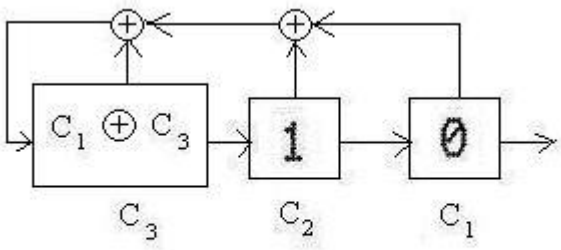
- La seconda cosa da fare è determinare i coefficienti della funzione di retroazione e cioè quali celle danno contributo al feedback. Per farlo, dobbiamo conoscere gli stati del registro in varie unità di tempo. All'inizio abbiamo il registro al suo stato iniziale e ancora non sappiamo nulla di quali sono le celle che danno contributo alla funzione.



Quindi all'inizio dobbiamo supporre che tutte le celle partecipino al feedback.

Procedendo… considero le celle con 1, quindi avrò:

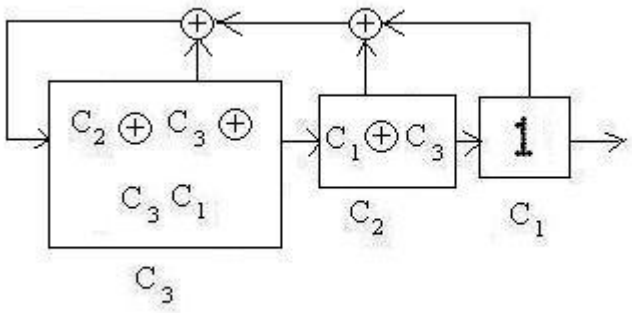
$C1 \cdot 1 + C3 \cdot 1 = C1 + C3$



Il contenuto dell'ultima cella sarà il risultato del feedback.
Per le altre celle il contenuto sarà determinato dal semplice shift a destra.

Quindi, dopo il secondo shift si avrà:

$C2 \cdot 1 + C3\,(C1 + C3) = C2 + C3 \cdot C1 + C3 \cdot C3 = C2 + C3 + C3 \cdot C1$



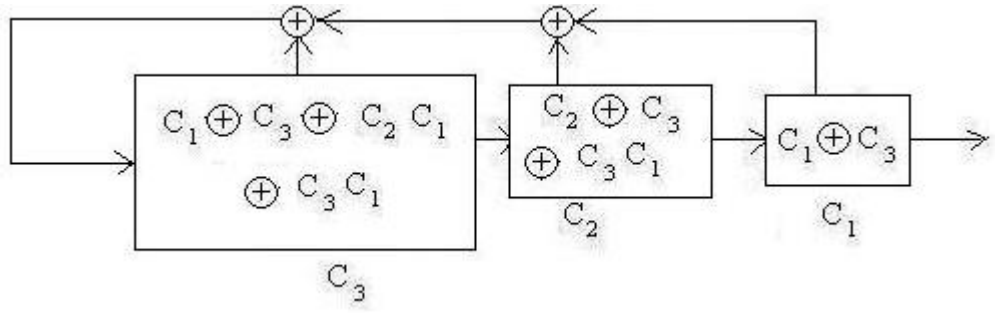
Dopo il terzo e ultimo shift si avrà:

$C1 \cdot 1 + C2\,(C1 + C3) + C3\,(C2 + C3 + C3 \cdot C1) =$

$C1 + C2 \cdot C1 + C2 \cdot C3 + C3 \cdot C2 + C3 \cdot C3 + C3 \cdot C3 \cdot C1 =$

$C1 + C2 \cdot C1 + 0 + C3 + C3 \cdot C1 =$

$C1 + C3 + C2 \cdot C1 + C3 \cdot C1$



Dall'ultimo scorrimento abbiamo ottenuto i rimanenti 3 bit della chiave gia conosciuta dal crittoanalista quindi possiamo scrivere queste 3 uguaglianze:

$C1 + C3 = 0$
$C2 + C3\,C1 + C3 = 0$
$C1 + C3 + C2 \cdot C1 + C3 \cdot C1 = 1$

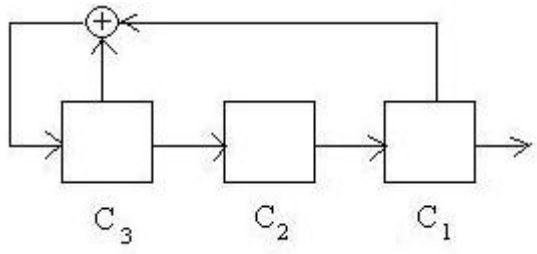
Da cui otteniamo che:

$C1 + C1 = 0$
$C2 + C1\,C1 + C1 = 0$
$C1 + C1 + C2 \cdot C1 + C1 \cdot C1 = 1$

Quindi: $C2 = 0$ perché $x \cdot x = 0$ e $x + x = 0$

Avrò così : C1 = 1 , C2 = 0 , C3 = 1

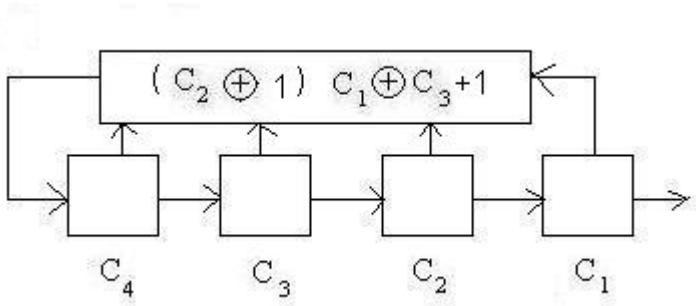
Allora la funzione di retroazione del registro che ha generato la chiave 101001 si può rappresentare più precisamente così:



Per concludere si può dire che:
si può forzare un registro a scorrimento lineare di lunghezza n (e quindi con periodo massimo di 2^n-1) se si conosce una sequenza di 2n bits successivi di testo in chiaro e i corrispondenti bits del testo cifrato.
Questa è una cosa incredibile se si pensa che una sequenza che si ripete dopo 1000000 bits quindi con un periodo dell'ordine di 2^{20}-1 bits può essere ricostruita conoscendo solo 40 suoi bits.
Questo scoraggia un po' l'uso di questi registri che comunque presentano molti vantaggi:

- Facilità e basso prezzo di realizzazione
- Produzione di sequenze pseudocasuali a grande velocità
- Sequenze ottenute danno risultati statistici molto buoni.

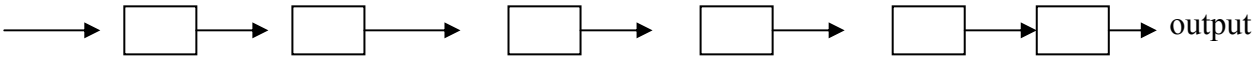
Per eliminare queste fragilità dal punto di vista crittografico sono stati introdotti i registri a scorrimento non lineari dove viene usata una funzione di retroazione più complicata (non lineare appunto). Infatti oltre alla somma viene utilizzata anche la moltiplicazione tra bits.
In figura è rappresentato un registro a scorrimento non lineare



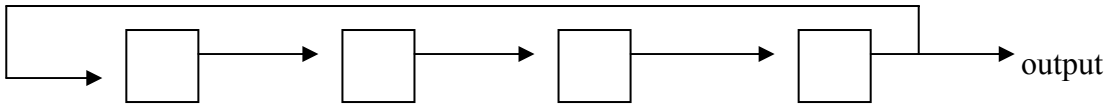
Spesso i registri a scorrimento non lineari sono realizzati combinando in modo non lineare registri lineari.
Comunque neppure quest'ultimi possono dare una sicurezza perfetta, ma sono sicuramente migliori di quelli lineari semplici.

I registri a scorrimento possono essere di due tipologie: **Lineari** e **Non lineari**. Questi sono conosciuti perché consentono una implementazione molto veloce, e producono sequenze con grandi periodi e buone proprietà statistiche se il polinomio della funzione retroattiva viene scelto correttamente.

In generale possiamo dire che un registro a scorrimento è formato da celle messe in serie, ognuna delle quali è in grado di contenere un bit, per cui la sequenza sulla quale si opera è una successione di 1 e 0.



Un registro ha lunghezza n e, la sequenza di zeri ed uno di cui si compone ne determina lo stato in quel momento, quindi possiamo dire che gli stati possibili sono 2^n.
La struttura è controllata da un clock che, ad intervalli di tempo regolari, fa passare il contenuto di una cella , seguendo il verso delle frecce, nella cella successiva: la cella più a destra manda fuori il suo bit e riceve il contenuto della cella precedente, questo fino all'ultima cella che cede il suo contenuto alla penultima. Ovviamente, cosi facendo, il bit dell'ultima cella non viene rimpiazzato, e dopo l'ennesimo clock il registro diventa vuoto. Si ha perciò necessità di riempire l'ultima cella, in modo da rendere ciclica la struttura.
Per fare ciò si potrebbe fare in modo di re-inviare il bit uscente dalla prima cella oltre che in output anche nell'ultima cella del registro, creando la struttura di tipo



soluzione però non molto astuta, in quanto cosi facendo, dopo un periodo P si ritorna allo stato iniziale, e questo è proprio quello che non vogliamo.
Per risolvere questo problema, l'ultima cella deve ricevere in input altri valori che sono dati creando una retroazione; i valori vengono calcolati elaborando la somma dei dati presenti in quel momento in alcune delle celle che compongono il registro scelte arbitrariamente, e, proprio perchè si effettua una somma questa soluzione è detta **Lineare** (anche perché molto semplice).
Questa tipologia di registri a scorrimento non viene usata molto proprio per la linearità, che li rende suscettibili ad attacchi algebrici;
Occorre pertanto inserire funzioni di retroazioni più complesse in modo da poter anche moltiplicare il contenuto di certi stadi: prendono cosi vita i registri con **Clock** e i registri **non lineari,** riguardo i quali non ci sono teorie, possiamo solo dire che la feedback function (retroattività) è molto complessa, e consiste in operazioni di complessità n che evitano ridondanze.

Registri nel campo informatico

Nel mondo informatico, un registro è un oggetto in genere di **32 bit** (termine anche usato per una generazione di computer nella quale si usa in genere un processore a 32 bit.

L'intervallo di valori interi che è possibile memorizzare in 32 bit parte da 0 fino a 4.294.967.295, o da -2147483648 fino a 2147483647 usando la codifica complemento a due, mentre per i 64 bit il range raddoppia.

Nota. La rappresentazione in complemento a due deve il suo nome alla regola secondo la quale la somma senza segno di un numero di n bit e del suo negato è pari a 2^n, e quindi il complemento (o negazione) di un numero x in complemento a due è pari a 2^n-x.

Registri a scorrimento (nel dettaglio).
Ogni registro a scorrimento lineare, prima di essere usato come generatore di sequenze pseudocasuali deve essere inizializzato, ovvero bisogna introdurre la sequenza di uno e di zero da

cui il generatore deve partire per generare le successive sequenze. Tale valore, assieme al polinomio di retroazione *f*(*x*), costituisce la chiave segreta che serve al mittente e al destinatario per ricostruire la sequenza pseudocasuale necessaria per le operazioni di cifratura e decifratura del messaggio in chiaro; quindi al variare di uno o di entrambi questi elementi, varia la sequenza degli output. Dunque, per cambiare la chiave, si può cambiare lo stato iniziale ed usare la stessa funzione di retroazione, oppure cambiare la funzione di retroazione o entrambi.

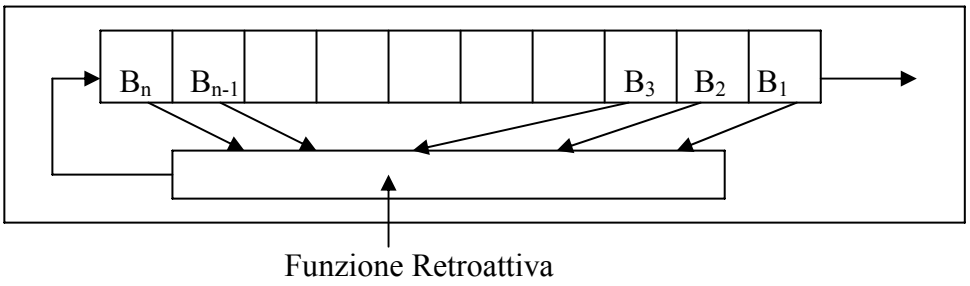
Esistono diverse implementazioni hardware dei registri a scorrimento, quella più utilizzata nell'ambito della crittografia è la tipologia S.I.S.O.
Ecco la classificazione dei registri a scorrimento secondo il tipo di caricamento dei dati:

1. S.I.S.O. (Serial Input Serial Output): l'ingresso e l'uscita dei dati avviene in modo seriale.

2. S.I.P.O (Serial Input Parallel Output): l'ingresso dei dati avviene in modo seriale, mentre l'uscita avviene in modo parallelo (tutti i bits contemporaneamente dopo un numero di shift adeguati).
3. P.I.S.O (Parallel Input Serial Output): l'ingresso dei dati avviene in modo parallelo, mentre l'uscita avviene in modo seriale.
4. P.I.P.O. (Parallel Input Parallel Output): l'ingresso e l'uscita dei dati avviene in modo parallelo.

Clock controller

Sui registri a scorrimento ci sono molte teorie, che nascono con la crittografia militare e l'elettronica. Un registro a scorrimento è formato da due parti, **il registro** e la **feedback function**, una scatola che contiene il cuore del sistema retroattivo.
Una possibile rappresentazione può essere:



Il registro è formato da una sequenza di bit , che vengono shiftati di una posizione ad ogni intervallo di tempo grazie ad un **controllore.**
Il controllore è lo strumento fondamentale per l'output dei dati, esso infatti scandisce il tempo per lo shit, e ci consente di utilizzare i registri non lineari, in quanto usando un clock irregolare si riduce il pericolo di attacchi. Per cui nella nostra figura si devono aggiungere altri componente:

- Un **registro di controllo** che genera una sequenza di interi non negativi a = $\{a_i\}i\geq 0$ e cicla con periodo $\prod$.
- Un **registro di generazione controllato da clock**, che ha un polinomio irriducibile nella funzione retroattiva, di grado m>1 e di ordine M.
- Il clock opera con la funzione di retroazione, che consiste in un polinomio.

Se indichiamo con b = (b(i)) $i \geq 0$ la sequenza di output prodotta da un registro quando viene cloccato regolarmente, si ha:
il primo elemento della sequenza è in posizione b(0) ed è pari al primo elemento dello stato corrente del registro puntato dal valore a_0 (prodotto dal registro di controllo), mentre l'ultimo prima dello shift è b(t-1) (con t lunghezza totale sequenza); il registro di controllo specifica un nuovo intero non

negativo a_t , il registro cloccato viene scalato a_t volte e produce l'output successivo, poi il registro di controllo scala di uno per la successiva iterazione.
Questa tecnica(basata sul clock) viene definita **forward clock control**.

L'elenco dei bit che fanno parte del polinomio di retroazione viene chiamato tap sequence.
Ad un registro a scorrimento di lunghezza n si associa come funzione di retroazione il polinomio $f(x) = a0 + a1x + ::: + an¡1xn¡1 + xn$ (4) cioè un polinomio di grado *n* (pari alla lunghezza del registro stesso), avente come coefficienti i coefficienti della funzione di retroazione. Tale polinomio è detto polinomio del registro.

(es. codice java)

Possibili attacchi
I registri a scorrimento lineari non forniscono sequenze pseudocasuali utili dal punto di vista crittografico. In fatti se un attaccante sa che il testo cifrato è stato ottenuto usando un registro a scorrimento di lunghezza *n* e riesce a scoprire 2*n* bit di testo in chiaro ed i corrispondenti 2*n* bit di testo cifrato, può ricostruire la chiave facilmente, perchè deve solo risolvere un sistema lineare di *n* equazioni in *n* incognite.

Def 1: Un clock irregolare si definisce forzato quando il suo intervallo è limitato da alcuni valori.

Si assume che la chiave segreta determini lo stato iniziale dei registro di generazione.

Un attacco consiste nella ricostruzione dello stato iniziale del registro generativo, partendo da un segmento della sequenza di output prodotta; pertanto si dovrà decidere quale stato iniziale porta alla massima probabilità di successo; per effettuare l'attacco quindi ci deve essere un punto di unione tra stringa di output prodotta dal clock irregolare, e l'output del generatore, quando cloccato regolarmente.
Per effettuare attacchi sono state proposte alcune tecniche basate su calcolo statistico:

- **Edit distance correlation attack**
- **Embedding correlation attack**

La base dell'attacco di tipo edit distance, si basa sulla misura della distanza tra due sequenze di lunghezza diversa, definita in modo da riflettere la trasformazione della sequenza di output prodotta dal registro di generazione in un output che chiamiamo u in base al modello statistico scelto.
Questa misura permette di fare una scelta statistica tra le varie possibilità, ed una ipotesi è accettata se, definendo Un un segmento di lunghezza n preso in modalità random dalla sequenza di output reale e Xm un segmento della sequenza generata di lunghezza m supponendo un possibile stato iniziale del registro, la distanza tra Un ed Xm è maggiore della soglia stimata sulla base del ragionamento scelto.
Ovviamente, questo ragionamento, si basa solo sulla edit distance che è troppo generica, per cui l'algoritmo risulta poco praticabile.

Nell'attacco di tipo Embedding l'obiettivo è quello di trovare tutti i possibili stati iniziali del generatore, così che per alcuni $m \geq n$ un segmento dato possa essere incluso nella sequenza di output di lunghezza m del generatore prodotta sotto un clock regolare ma l'attacco ha successo se ci sono solo pochi di questi stati iniziali.
Per verificare se l'inclusione sia possibile, si può usare un algoritmo di corrispondenza diretta (direct matching algorithm) per l'inclusione forzata o si possono usare algoritmi per calcolare la distanza. L'inclusione è possibile se e solo se la distanza è uguale a m – n.

E' ovvio che embedding attacks generalmente non sono ottimali dal momento che non fanno uso della distribuzione probabilistica della sequenza di controllo.

Attacco correlato: Un pregio dei registri è l'*immunità da correlazione cioè* la probabilità che sussista una relazione privilegiata tra l'uscita del generatore di sequenze pseudocasuali e l'uscita di uno dei suoi LFSR interni. L'immunità da correlazione è importante perché, studiando il comportamento otteniamo informazioni circa le sequenze interne. Usando queste informazioni ed altre relazioni, riusciamo a forzare il generatore.

ALGORITMI CHE UTILIZZANO REGISTRI A SCORRIMENTO

A5

L'A5 è un cifrario di flusso, utilizzato principalemente per criptare trasmissioni via cellulare (GSM).
Questo algoritmo infatti viene utilizzato a tutt'ora per criptare il collegamento tra un terminale(telefonino) e la base (ricevitore).
Mentre la trasmissione vocale tra un cellulare e l'altro non viene decrpittata.
L'A5 consiste in tre registri a scorrimento linari, la cui lunghezza è relativamente 19, 22 e 23. Tutti i polinomi del feedback sono "sparsi".
L'ouput è fornito dallo XOR di questi tre registri. Inoltre questo algoritmo utilizza un clock variabile.
Un attacco più banale richiede 2 40 tentativi.
Infatti l'idea di base dell'A5 è molto buona ed è un algoritmo molto efficiente, infatti ha passato tutti i test statistici. L'unico difetto è probabilmente che i suoi registri sono troppo corti, ma basterebbe una leggera modifica o variante dell'A5 per creare un algoritmo più sicuro.

Huges XPD/KPD

Questo algoritmo è stato progettato dalla Huges Aircraft Corporation nel 1968.
Venne applicato su alcune apprecchiature per uso militare come radio, localizzatori
Etc…
Inizialmente si chiamaza XPD e aveva l'EPD (Dispositivo Esportabile di Protezione), successivamente fu rinominato KPD (Dispositivo Cinetico di Protezione).
Questo algoritmo usa un registro a 61 bit con 2 10 differenti polinomi primitivi nel FEEDBACK, approvati dalla NSA (National Security Agency). La chiave seleziona uno di questi polinomi che sono allocati da qualche parte nella memoria, assieme allo "stato iniziale" del registro.

La chiave ha 8 differenti filtri non lineari, che producono 6 "TAP".
Ogni "TAP" produce un bit, questi si combinano per formare un byte che viene usato per criptare o decriptare un flusso di dati.

NANOTEQ

Il nome NANOTEQ deriva da una compagnia elettronica del Sud Africa.
Inizialmente questo algoritmo veniva utilizzato dalla polizia per cifrare le loro trasmissioni, soprattutto quelle via fax.

Questo algoritmo usa un registro a 127 bit con un polinomio fisso nel feedback. La chiave, in questo caso, è lo stato iniziale del registro.
I 127 bit del registro sono ridotti ad un singolo flusso di bit, questo comporta l'utlizzo di 25 celle. Ciascuna cella ha 5 input de un solo output. Ciascun input della funzione è messo a XOR con alcuni bit della chiave. Questi bit infatti sono scelti da chi vuole criptare il messaggio, ovvero cambiano in base alla particolare implementazione.
A questo proposito esiste una permutazione segreta.
Questo dispositivo è solo disponibile via hardware e non abbiamo altre informazioni,visto che è un algoritmo molto recente.
Un crittoanalista: Ross Anderson ha iniziato solo ora a fare i prmi passi per analizzarlo.

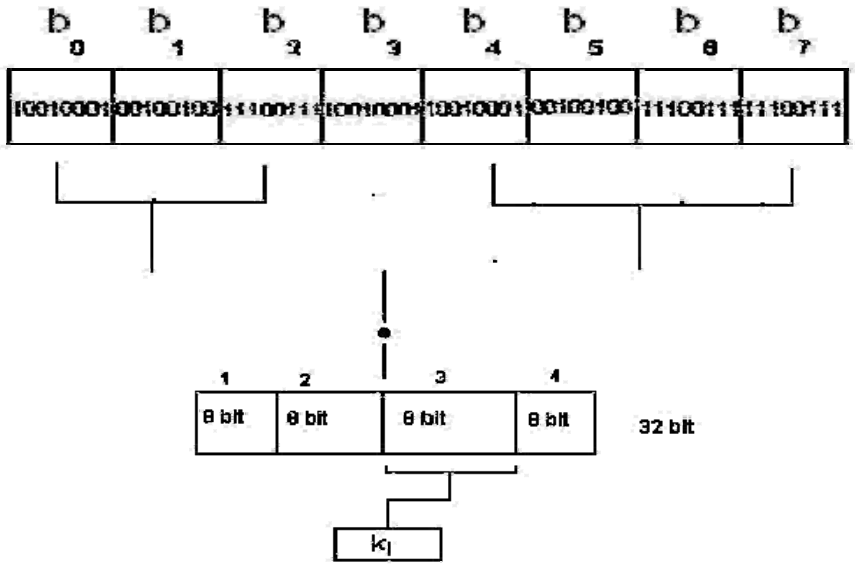
RAMBUTAN

Il Rambutan è un algoritmo di origine britannica, che è stato progettato dalla GCHQ(Gruppo della Sicurezza nella Comunicazione Elettronica).
Questo algoritmo fu approvato per la protezione di materiale ritenuto "Confidential".
Questo algoritmo è a tutt'ora segreto e il chip non è disponibile in commercio.
Il Rambutan ha una chiave a 112 bit e può operare in 3 diversi modi: ECB,CBC e CFB a 8 bit.
Probabilmente è un registro per i cifrari di flusso.
È composto da 5 registri, ognuno di lunghezza di circa 80bit.
I polinomi nel feedback sono "sparsi" e ottengono solo 10 "TAP" ognuno.
Ogni registro a scorrimento fornisce 4 input a una funzione non lineare complessa e molto grande.

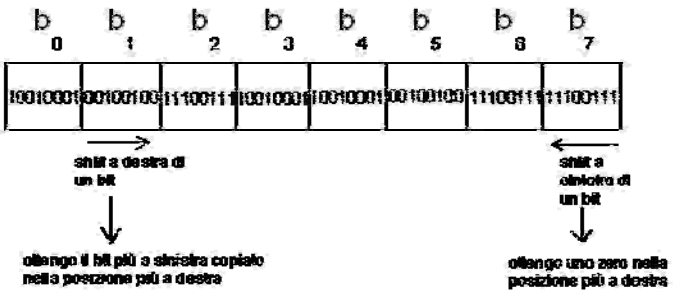
GIFFORD

Questo algoritmo fu inventato appunto da David Gifford che lo utilizzava per criptare le notizie che circolavano nell'area di Boston tra il 1984 fino al 1988. L'algoritmo ha un singolo registro a 8 byte Indicati con: b0,b1,b2….,b7.
La chiave è lo stato iniziale del registro. Questo algoritmo lavora in OFB(output feedback).
Per generare un byte della chiave (Ki), si deve concatenare b0 e b2, b4 e b7 e moltiplicare tra loro i membri risultanti per ottenere un numero a 32 bit. Ora, il terzo byte dalla sinistra del risultato è proprio Ki.
Per aggiornare il registro, invece, si prende b1 e si shifta a destra di un solo bit. Questo comporta che ci sia ci sia una copia subito a destra, del bit più a sinistra di b1 nella posizione accanto.
Successivamente di prende b7 e si shifta di 1 a sinistra, questo comporta che ci sia uno zero nella posizione più a sinistra di tutte.
Si fa uno XOR tra: b1 modificato, b7 modificato e b0.
Ora, si shifta tutto il registro di un byte e si mette il byte dello XOR precedente all'inizio del registro.

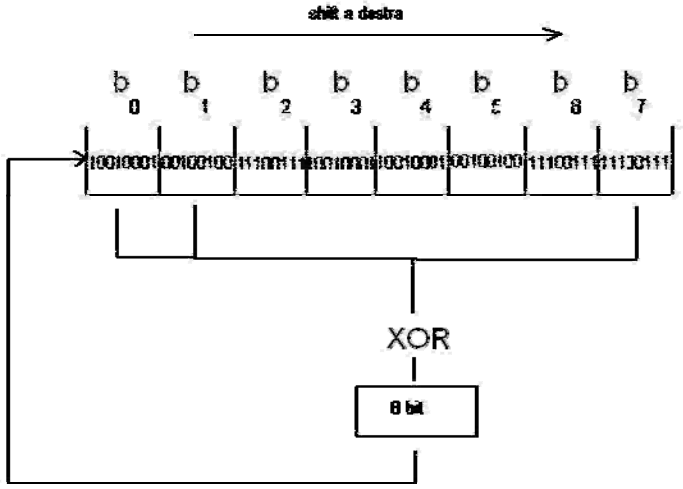
Come trovare un byte della chiave:



Aggiornare il registro(1):



Aggiornare il registro (2):



Questo algoritmo rimase sicuro per tutta la sua vita, ma fu rotto nel 1994.

PKZIP

Roger Schalfly progettò la codifica per l'algoritmo utilizzato nel noto programma di compressione dati: PKZIP.
Questo algoritmo è un cifrario a flusso e cripta un byte alla volta.
L'algoritmo utilizza tre variabili a 32bit, inizializzate come segue:
K0=30541989
K1=591751049
K2=878082192
Inoltre ha una chiave a 8 bit chiamata K3 che è derivata da K2 secondo alcune compesse operazioni.
Sfortunatamente, non è molto sicuro. Un attacco richiede da 40 a 200 byte per conoscere il testo in chiaro e ha una complessità di circa 2 27.
Ma non è un algoritmo utilizzato per nascondere informazioni, ma solamente per comprimere dati.

REGISTRI A SCORRIMENTO NON LINEARI

È facile immaginare una sequenza più complicata nel feedback di quella usata nei registri a scorrimento lineare.
Il problema è che non esiste una teoria matematica per analizzarle.
In particolare ci sono alcuni problemi per le funzioni non lineari:

- Ci possono essere più zeri che uni, o meno "RUN" di quanti ci si aspetti, nella sequenza di output
- Il periodo massimo di una sequenza può essere più corta di quanto ci si aspetti.
- Il periodo della sequenza, potrebbe essere differente per differenti valori di partenza.
- La sequenza potrebbe sembrare random per un po' e poi andare a collassare in un singolo valore.

Per quanto riguarda invece i pregi, non ci sono teorie per analizzare i feedback non lineari con certezza, e ci sono pochissimi mezzi che un crittoanalista può utlizzare per decriptare un messaggio.
Quindi si possono usare, ma con molta cautela.

In pratica, in un registro a scorrimento non-lineare, la funzione di feedback può essere qualsiasi cosa tu voglia.