Cifra Sequenciais

MICEI-MSDPA

José Carlos Bacelar Almeida
(jba@di.uminho.pt)

Cifra One-Time-Pad

- Já estudamos o facto da cifra OneTimePad (Vernam) oferecer garantias de confidêncialidade!

$C_i = T_i \oplus K_i$

MAS:

$K_i = T_i \oplus C_i$

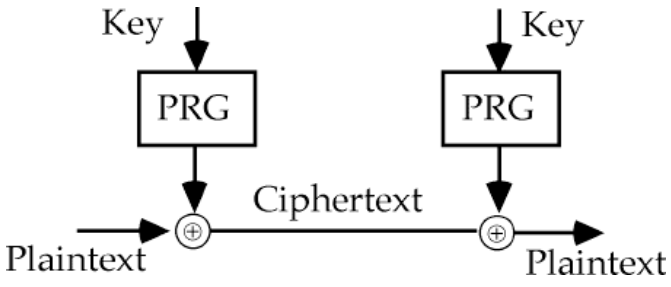
$C^1_i = T^1_i \oplus K_i$ e $C^2_i = T^2_i \oplus K_i$ determina que $T^1_i \oplus T^2_i = C^1_i \oplus C^2_i$

Problemas da cifra OneTimePad

- Sequência de chave deve ser verdadeiramente aleatória e de comprimento igual à mensagem.
- Sequências de chaves nunca devem ser reutilizadas. Um ataque com texto limpo conhecido é trivial, e mesmo só dispondo dos criptogramas podemos retirar muita informação útil sobre o texto limpo.
- Não promove a difusão da informação no criptograma. Se dispusermos de informação sobre uma mensagem podemos "trocar" os bits pretendidos dessa mensagem (ou seja, a integridade da mensagem tem de ser garantida por outros meios).

Cifras sequenciais

- A ideia base consiste em "aproximar" a cifra OneTimePad por intermédio de um gerador de chaves (que produz uma sequência de chave a partir de uma chave de comprimento fixo).



Características

- Processam o texto limpo "símbolo a símbolo" (bit a bit, carácter a carácter, digito a digito...).
- Tendem a ser muito eficientes e facilmente implementáveis em hardware.
- O processo de geração da sequência de chave tem de ser reprodutível – pode por isso ser visto como uma máquina de estados finita. Como consequência directa temos que a sequência tem necessariamente de ser cíclica. Diz-se que o **período** é o comprimento da sequência antes de se começar a repetir.

Critérios para o desenho de Cifras Sequenciais

- Período deve ser tão grande quanto possível (sempre maior do que a mensagem a transmitir).
- Sequência de chave deve ser:
  - **pseudo-aleatória**: propriedades estatísticas análogas a uma sequência "verdadeiramente" aleatória.
  - **imprevisível**: dado um segmento inicial não deve ser possível prever a sua continuação.
- Outras propriedades...
  - Propagação de erros e/ou sincronismo
  - ...

Cifras Auto-Sincronizáveis

- Cada bit da chave é calculado a partir dos últimos *n* bits do criptograma (e da chave, naturalmente).

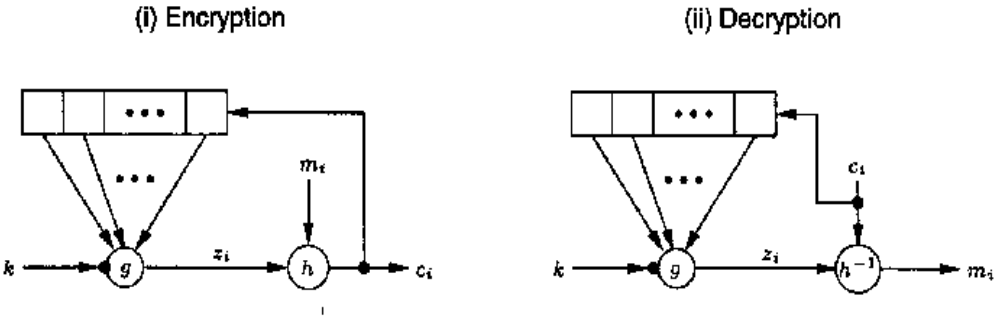


Figure 6.3: *General model of a self-synchronizing stream cipher.*

Características das Cifras Auto-Sincronizáveis

- Introduz-se um prefixo de *n* bits aleatórios no texto limpo para permitir sincronização da recepção.
- Ao fim de *n* bits a decifragem sincroniza (após erro de transmissão; omissão/inserção de bits no criptograma).

- Problemas:
  - Vulnerável a ataques por repetição (o intruso pode reenviar uma porção do criptograma).

Exemplo de uma cifra Auto-Sincronizável

$C_i=(C_{i-1}\oplus K_1)\oplus(C_{i-2}\oplus K_2)$

$K_1=1 \quad K_2=0$

Chave gerada	1	0	0	0	0	1	1	1	1	0	1	1	1	1	0	0	0
Mensagem	1	1	1	0	0	1	1	1	0	1	0	0	0	1	0	1	0
Criptograma	0	1	1	0	0	0	0	0	1	1	1	1	1	0	0	1	0

Criptograma	0	1	1	0	0	1	0	0	1	1	1	1	1	0	0	1	0
Chave gerada	1	0	0	0	0	1	0	0	1	0	1	1	1	1	0	0	0
Mensagem	1	1	1	0	0	0	0	0	0	1	0	0	0	1	0	1	0

Cifras Síncronas

- A sequência de chave é independente do texto limpo/criptograma.

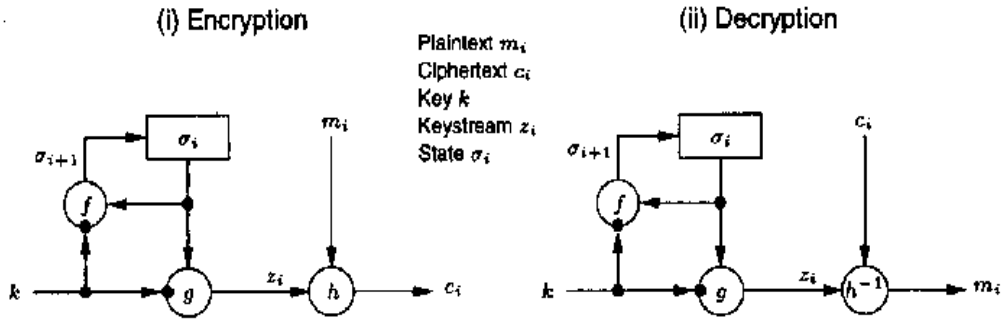


Figure 6.1: *General model of a synchronous stream cipher.*

Características das Cifras Síncronas

- A perca/inserção de bits no criptograma determinam a "perca de sincronismo". Ao decifrar, toda a mensagem a partir desse ponto é corrompida.
- Erros (alterações de bits) só alteram a posição correspondente da mensagem original.
- A chave (parâmetro de segurança) pode afectar:
  - A função *f* que determina o próximo estado – *Output Feedbak Mode.*
  - A função *g* de saída – *Counter Mode.*
  - Ambas...

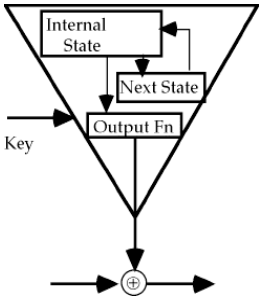
Sequências Pseudo-Aleatórias

- Critérios de *Golomb*
  - A diferença no número de 1s e de 0s deve ser tão pequeno quanto possível.
  - Quando particionamos a sequência em sub-sequências de símbolos repetidos (*runs*), devemos encontrar um número de *runs* de comprimento *l* dado por $r(l)=2^{-l}*r$ (se $2^{-l}*r>1$), onde *r* é o número de *runs*.
  - A auto-correlação deve ser um valor constante para qualquer desvio diferente de 0 (mod p).

$$C_p(\tau) = \frac{1}{p}\sum_{i=1}^{p} x_i x_{i+\tau}$$

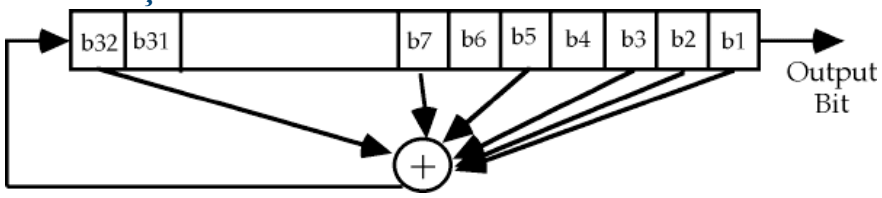
Teoria das Máquinas de Estados Finitos

- O problema de construção de sistemas para produção de sequências pseudo-aleatórias dispõe já de um conjunto de resultados importante...
- Os *Linear Feedback Shift Registers (LFSR)* constituem uma das principais abordagens ao problema, com um manancial importante de resultados associados.



LFSRs

- Dispositivo "síncrono" (controlado por um "relógio") um os bits são deslocados (produzindo um bit de saída) e o bit e entrada é determinado por uma função linear.



- A um LFSR podemos associar um polinómio (mod 2) onde os coeficientes não nulos correspondem aos bits utilizados no *feedback*.
  - E.g. $x^{32}+x^7+x^5+x^3+x^2+x+1$

Alguns resultados sobre LFSRs

- Um LFSR dispõe de período máximo (2^n-1) se e só se o polinómio que lhe é associado for primitivo. Além disso, a sequência resultante satisfaz os critérios de *Golomb*.
- ...mas isso não determina que sejam geradores de chaves satisfatórios...
  - são apenas necessários *2*n* bits da sequência para reconstruir o LFSR que lhes deu origem...
- Necessitamos então de "esconder" a estrutura matemática forte que está subjacente.

Combinação de LFSRs

- Solução típica consiste em agregar diferentes LFSRs de uma forma que obscureça a sua estrutura.
  - Uma função para a combinação de diferentes LFSRs.

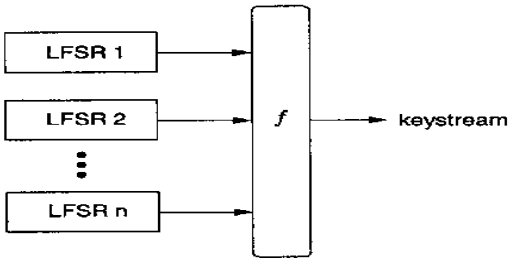


Figure 6.8: *A nonlinear combination generator. f is a nonlinear combining function.*

  - Função deve ser, de preferência, não linear para dificultar ataques por correlação (onde se extrai informação da saída para derivar informação de cada LFSR)

Combinação de LFSRs (cont.)

- Utilização de LFSRs para regular a cadência de outros.

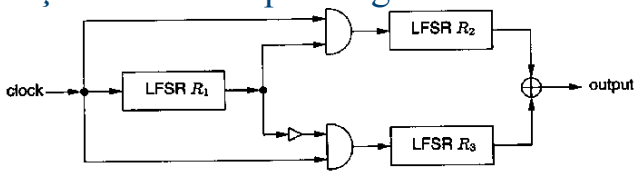


Figure 6.12: *The alternating step generator.*

- Utilização de LFSRs para filtrar a saída de outros.

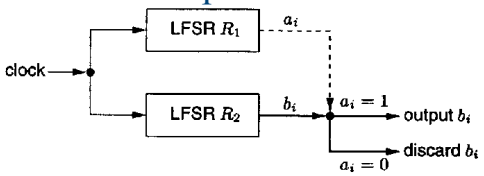
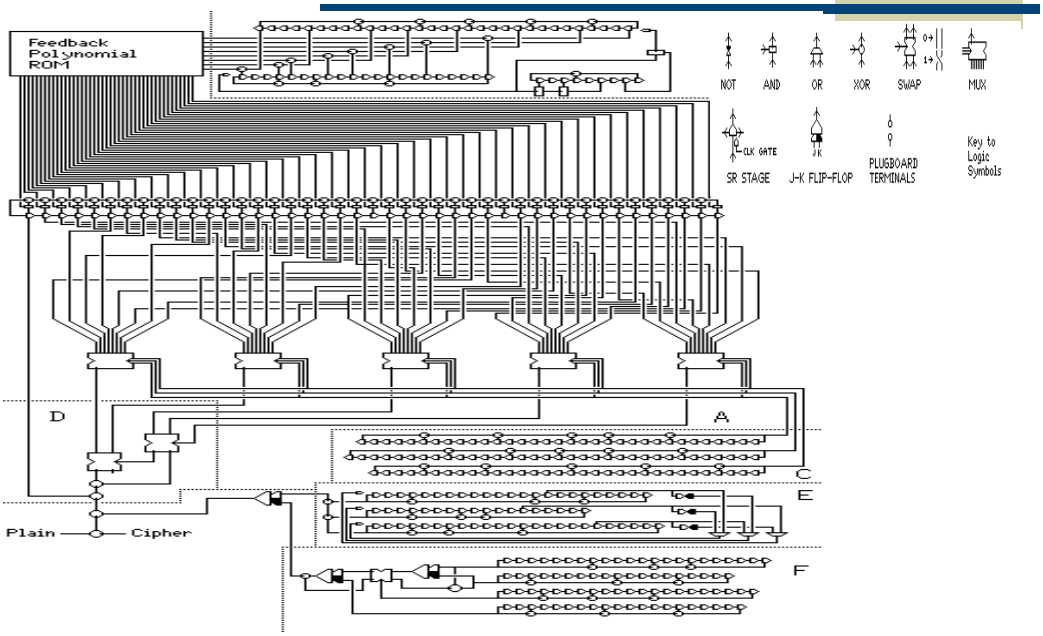


Figure 6.13: *The shrinking generator.*

- Multiplexers; Flip-Flops; etc.; etc.

Um exemplo...



A5 (A5/1; A5/2)

- Cifra utilizada no Standard Europeu GSM de comunicações móveis.
- Utiliza três LFSRs (de 19, 22 e 23 bits).
- Já quebrado... (2^{40})

- Vulgarmente reconhecido como um "bom desenho" mas *propositadamente fraco* em termos de segurança (registos pequenos e polinómios esparsos)

RC4

- Cifra desenvolvida por *Ron Rivest* (RSA Labs).
- Originalmente "trade secret" mas, por engenharia reversa, foi descoberto o algoritmo e divulgado por um *post* anónimo na *newsnet*.
- Vocacionado para ser executado em *Software* com operações ao nível do *byte*.
- Admite chaves de comprimento variável (até 2048 bit).
- Opera em *Output Feedback Mode*.
- Cerca de 10 vezes mais rápido do que o DES.

RC4 (cont.)

C code for generator for RC4

```
/* SIZE is (1<<ALPHA) = (1 times 2 to the 8th) = 256.
 * ind(x) is the low order 8 bits of x, or x mod 256.*/
#define ALPHA (8)
#define SIZE (1<<ALPHA)
#define ind(x) (x&(SIZE-1))
static void rc4(m,r,aa)
    int *m; /* Memory: array of SIZE ALPHA-bit terms */
    int *r; /* Results: the sequence, same size as m */
    int *aa;/* Accumulator: a single value */
 { register int a,x,y,i;
    a=*aa;
    for (i=0; i<SIZE; ++i)
    { x=m[i];
      a=ind(a+x);
      y=m[a];
      m[i]=y;
      m[a]=x;
      r[i] = m[ind(x+y)];
    }
    *aa=a;
  }
```

Geradores Computacionalmente Seguros

- Segurança do gerador da sequência aleatória é estabelecida com base num argumento de "Complexidade" (e.g. utilização de uma função de sentido único).
- Do ponto de vista de eficiência computacional, tornam-se naturalmente muito mais lentos do que os apresentados...
- Exemplos (ver bibliografia): *Blum, Blum and Shub; RSA; etc.*

Referências

- Stream Ciphers – *M. J. Robshaw*, RSA Labs TR-701, 1995.
- Applied Cryptography – *Bruce Schneier*.
- Basic Methods of Cryptography – *Jan C. A. van der Lubbe*, Cambridge Press, 1997.