Algebraic Attacks on

Stream Ciphers

Nicolas T. Courtois

University College London, UK

Part -1

Stream Ciphers

Stream Ciphers:

No clear frontier between Block Cipher and Stream Cipher Encryption…

A different philosophy of encryption:

- Don’t think about encrypting messages.
- More like encrypting a channel.

Stream Ciphers:

Possible Advantage: The adversary does not know that the message even exists.

[message + length]
[Cryptography + Steganography]

Used in Telecommunication, Military, Government, Diplomatic Applications.

(Traffic analysis is very important, who talks to whom and when, may be more important than knowing the plaintext !!!)

Block Ciphers:

- Usually Stateless and Deterministic.
- Encrypt one block of data at a time.

Stream Ciphers:

- Have a state that changes with time.
- Encrypt anything from one bit to long streams of data. (in a sense also 0).

Self-Synchro [Asynchronous] Stream Ciphers:

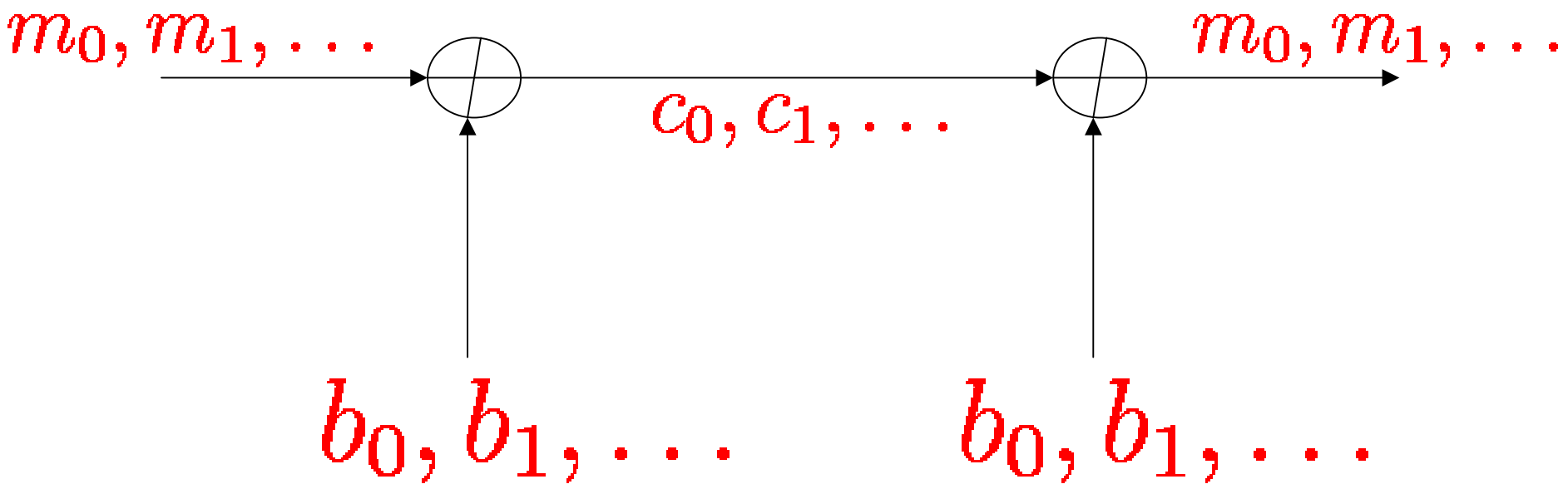
- The keystream depends on the key and on a fixed number of ciphertext bits !
- Self-synchronising: can re-establish an interrupted transmission.
- Very close to block cipher in both design and cryptanalysis. Hard to design and protect against chosen plaintext attacks….

Synchronous Stream Ciphers:

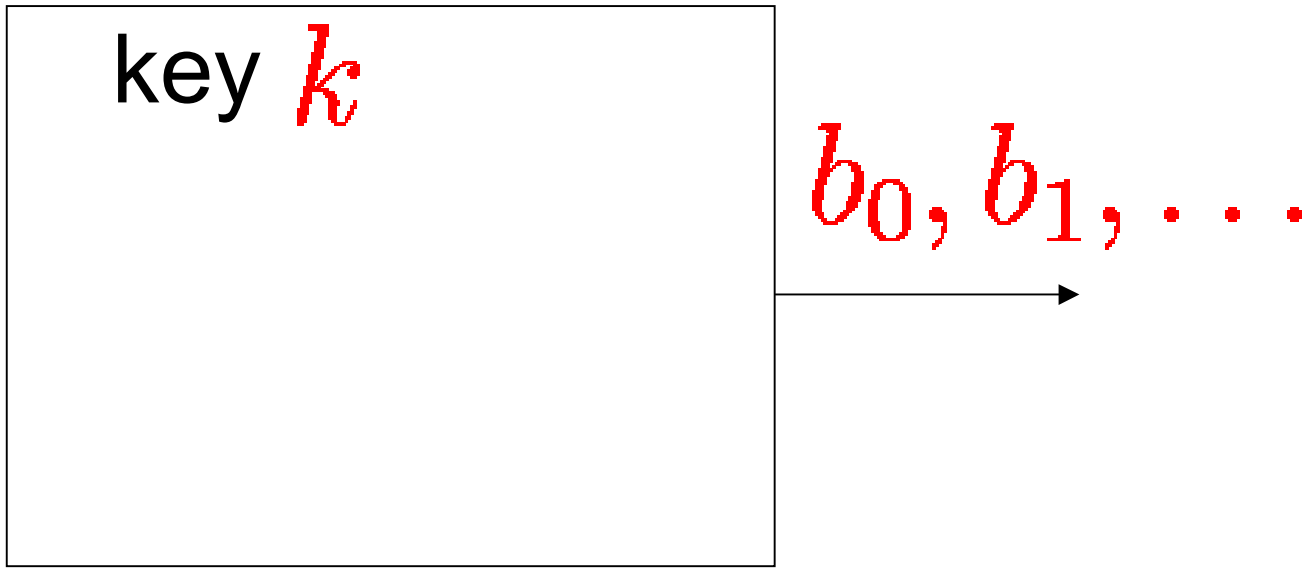
- Like a pseudo-random generator.
- Generates a keystream independent of the plaintext and XORed to the plaintext.

Synchronous Stream Ciphers:

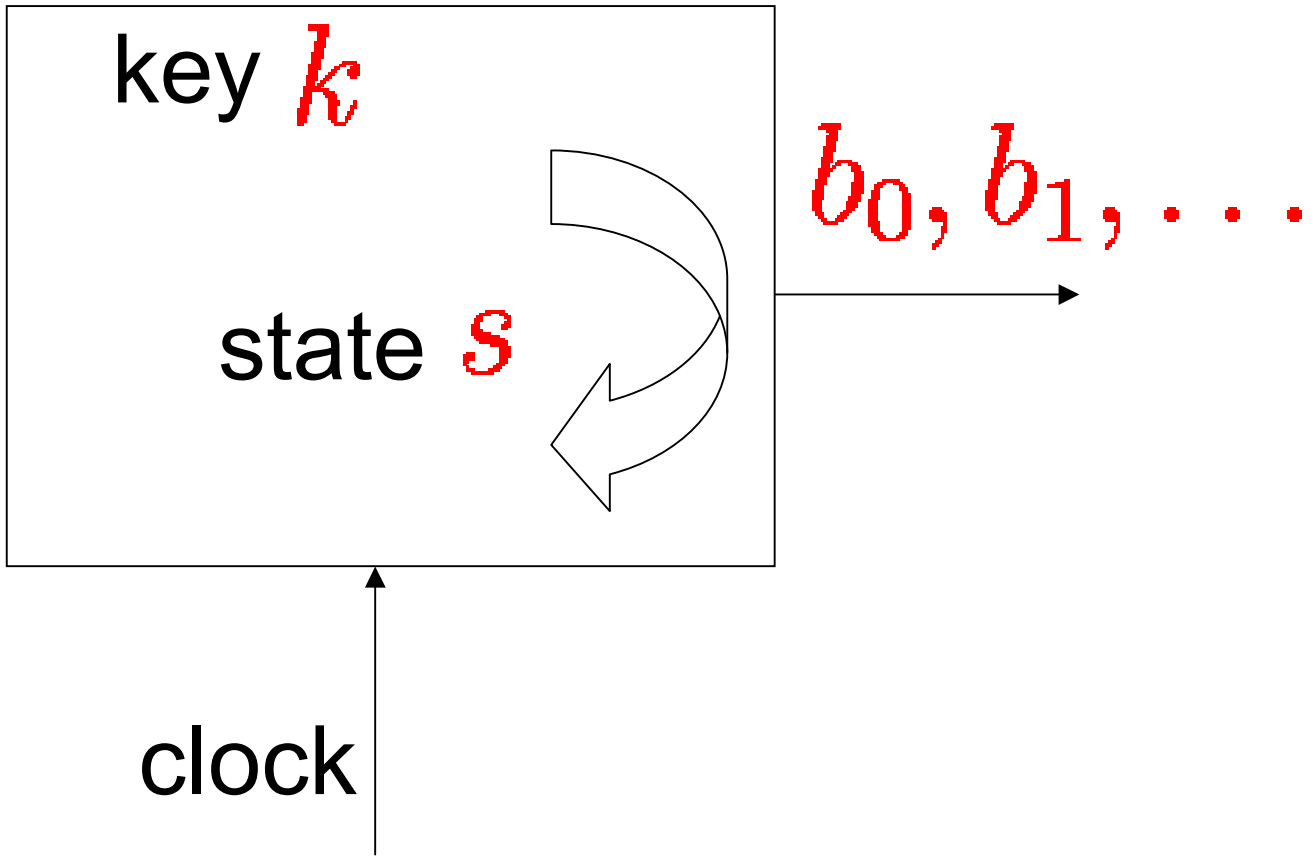
- "One-time pad" with pseudorandom sequence:



Keystream Generator:

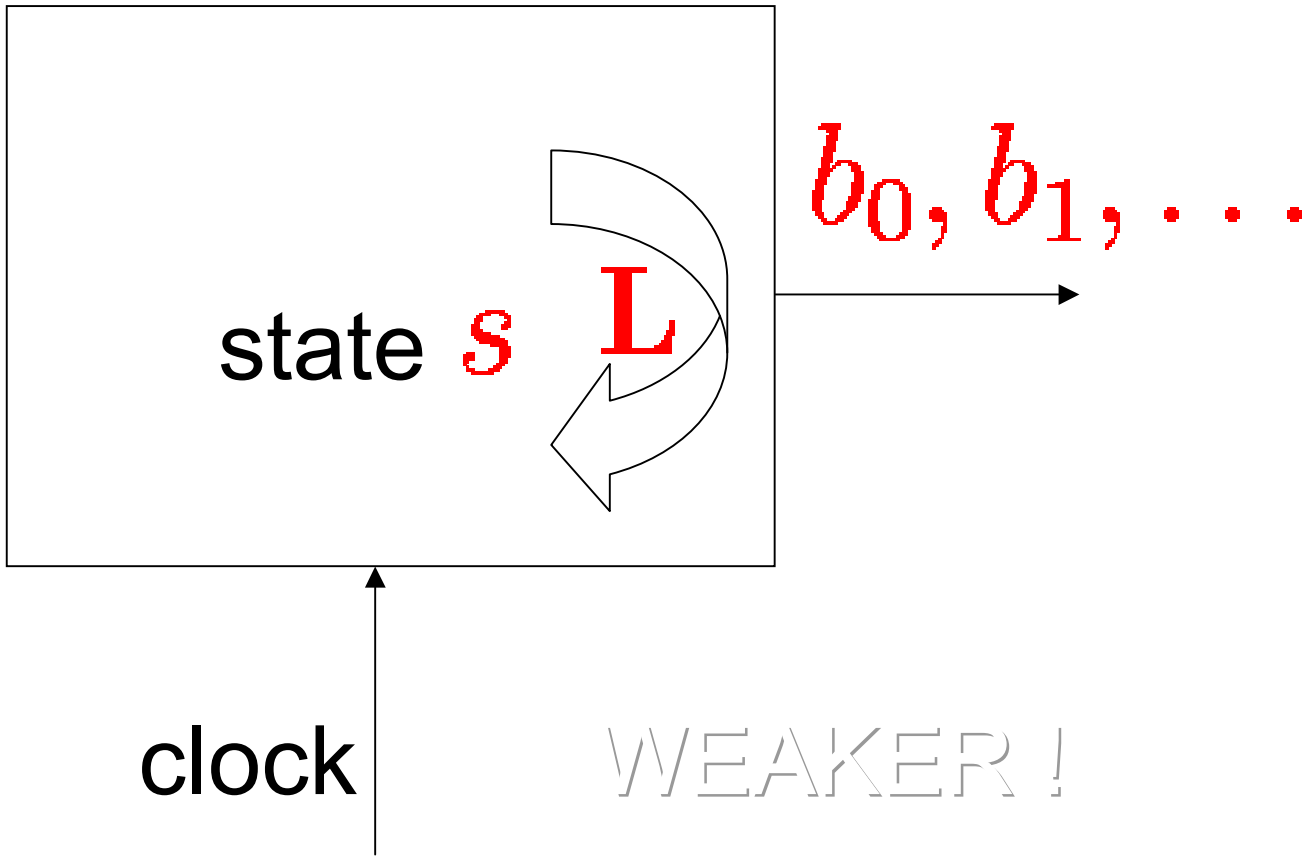


Keystream Generator:



Recursive Keystream Generator:

$$s^{(0)} = k$$

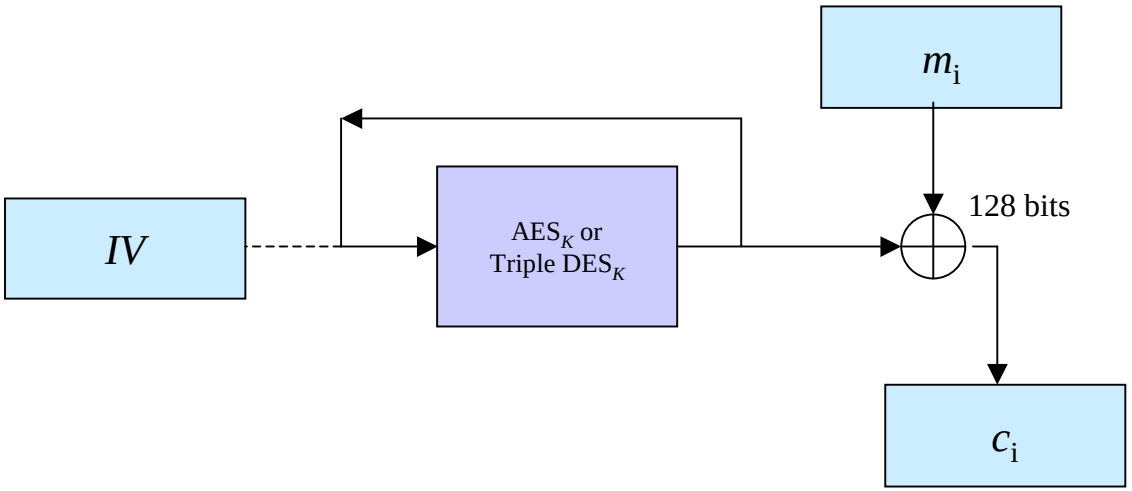


WEAKER !

OFB (Output FeedBack) mode

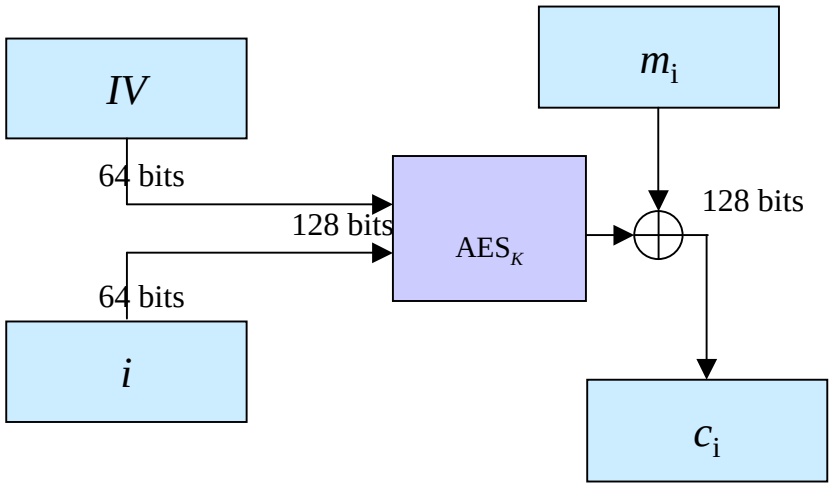
Makes a synchronous stream cipher

(out of a block cipher).



CTR = Counter mode

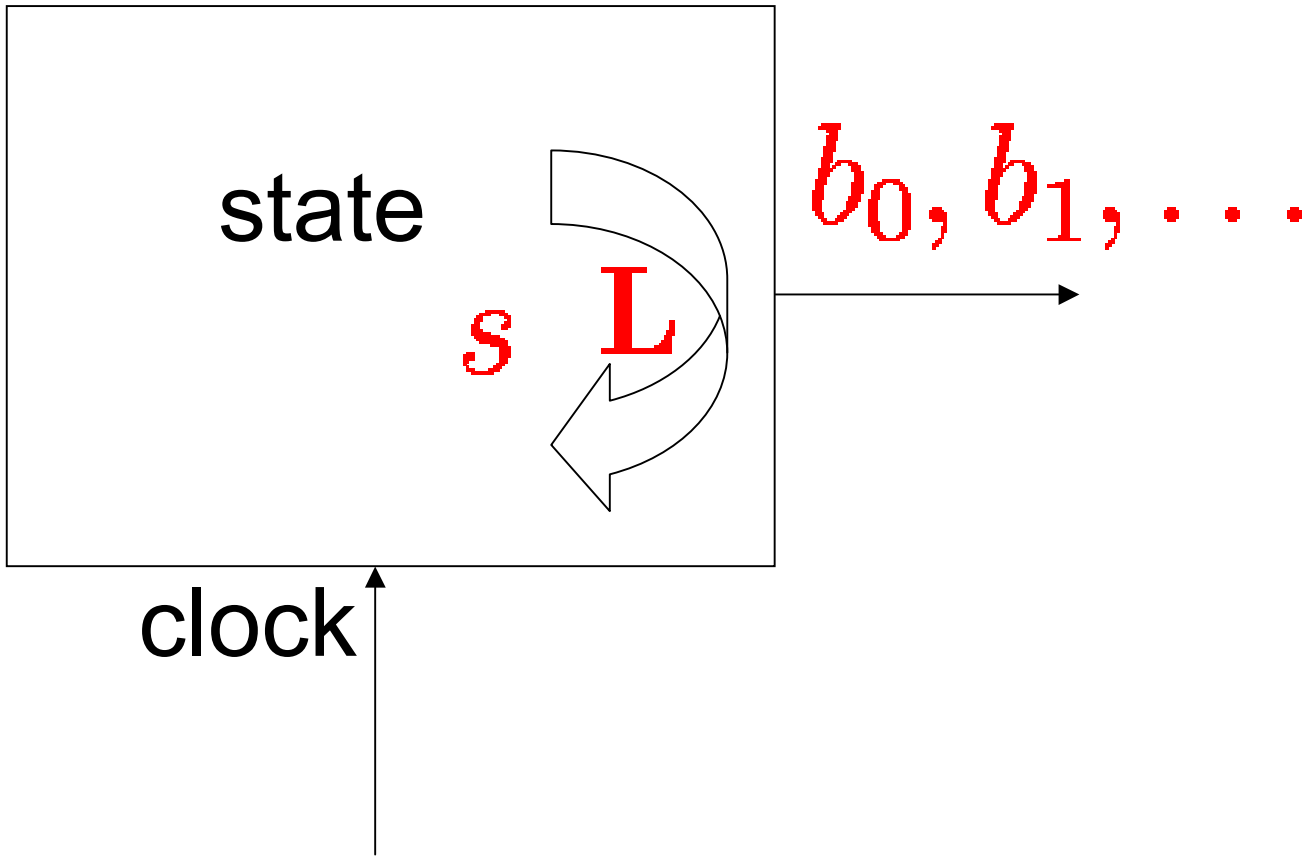
Also makes a synchronous stream cipher (out of a block cipher). There are several versions, one example (recommended !) with AES:



Recursive + Linear:

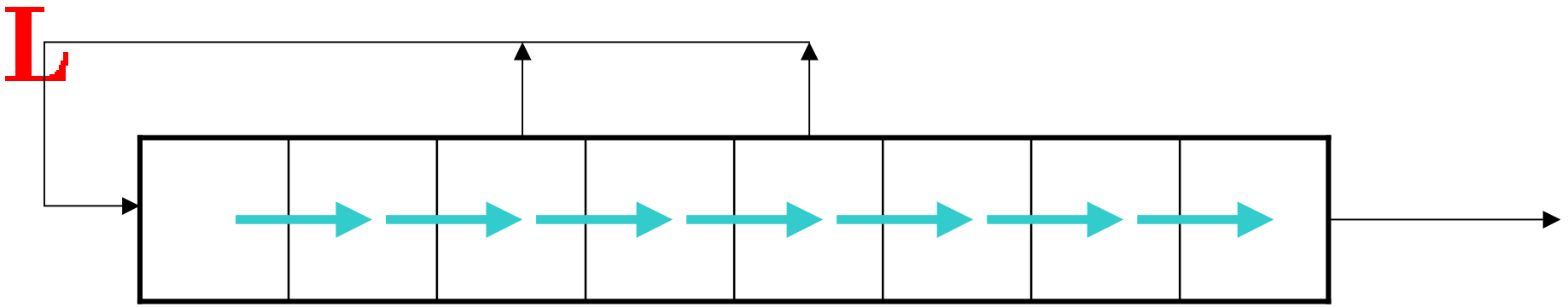
$s^{(0)} = k$

$\mathbf{L}$ - multivariate linear



Fast + Recursive + Linear:

One new bit a time…



$b_0, b_1, \ldots$

LFSR

(Linear Feedback Shift Register)

LFSR

- Hard to imagine faster generators of sequences.
  - Lots of “nice” properties…
- Not secure as it is. (2n bits -> broken).

- Idea: combine outputs of several LFSR (or internal bits of one LFSR).
- FILTERS/COMBINERS – no difference for us:
  - stateless – Boolean functions
  - stateful – Combiners with memory

Part 0

Stream Ciphers, Algebraic Equations & Algebraic Attacks

Scope:

Regularly Clocked[1] Stream Ciphers with Linear Feedback[2]:

(one or several LFSRs, or cellular automata…)

1. a bit of decimation/irregular clocking can also be tolerated…
2. a “small” non-linear state/memory can be tolerated…

- # [Purely] Algebraic Attacks:

Attacks that work by 1) writing 2) solving by some elimination techniques a system of algebraic equations.

- # Multivariate Cryptanalysis:

If these equations are multivariate polynomials…

- # Non-Linear Cryptanalysis:

1. Generalisations of Linear Cryptanalysis (GLC, BLC, etc.)
2. Other attacks on ciphers with very high non-linearity.

Not exactly synonymous, many attacks are all these, some only one of them…

Algebraic Attacks:

Solving systems of equations…

Favorite Equations:

Multivariate polynomials over a finite field, e.g. GF(2).

$$\begin{cases} 1 = x_1 + x_0x_1 + x_0x_2 + \dots \\ 0 = x_1x_2 + x_0x_3 + x_7 + \dots \\ \vdots \qquad\qquad \vdots \end{cases}$$

Why Algebraic Attacks:

Breaking a « good » cipher should require:

"as much work as solving a system of simultaneous equations in a large number of unknowns of a complex type"
[Shannon, 1949]

Common belief: large systems of equations become intractable very easily.

However…

However, what makes the problem hard is not the number of variables, but the balance between the number of equations and the number of monomials:

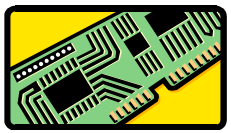
- The XL method:
  [Shamir, Patarin, Courtois, Klimov, Eurocrypt'2000]
- The XSL variant:
  [Courtois, Pieprzyk, Asiacrypt'2002]

Consequence: systems that are overdefined, sparse, or both, turn out to be much easier to solve than expected.

Threat 1: Overdefined Systems

Most cryptographic security relies on the hardness of largely overdefined problems: Much more information than necessary: great many plaintexts, message and signature pairs, etc..

- Public key cryptography: Solution: provable security: each utilization of the cryptographic scheme does not leak useful information.
- Secret key cryptography: Yet little provable security. It is also in secret key cryptography that the problems become the most overdefined: huge amounts of data encrypted with one key, fast hardware. Especially true for stream ciphers.



Threat 2: Sparsity

Most cryptographic schemes (for practical reasons) have a simple algebraic description.

Usually leads to a sparse system of equations.

- In software, large tables may be used.
- In hardware, the number of gates should be small, $\Rightarrow$ gives a simple description with simple Boolean polynomials. Especially true for stream ciphers.

Threat 3: Linear Feedback

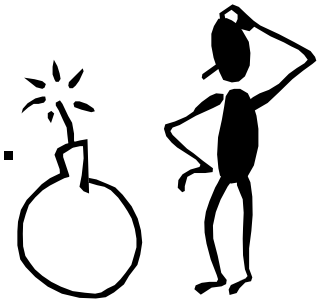
Linear feedback (e.g. LFSRs)

state = L(prev. state)

multivariate linear function

So what ?

Linear Feedback is Dangerous



It preserves the degree of the equations.

[Courtois 2003]:

If one can relate state bits and outputs bits by only one multivariate equation of low degree without extra variables then:

=> the cipher is broken in polynomial time !

Common Opinions on Stream Ciphers

"Most real life designs centre around LFSRs combined by a non-linear Boolean function."

"State of the art in generic stream ciphers cryptanalysis can be summarized as follows: correlation and fast correlation attacks."

[Eric Filiol, Decimation Attack of Stream Ciphers, eprint.iacr.org, 2000]

Stream Ciphers…

Are NOT as secure as you think…

Common belief:

Ciphers with linear feedback (LFSR, etc…)

can be made secure using

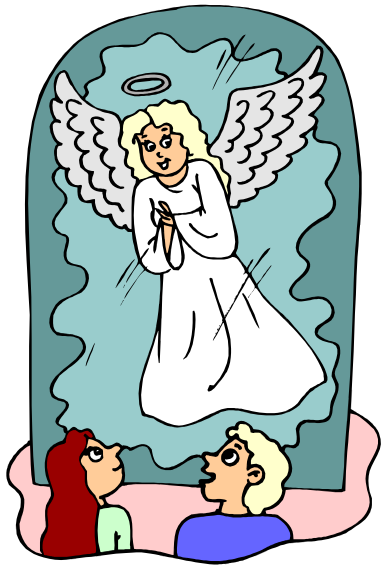
highly non-linear Boolean functions.

The Tale of "Good" Boolean Functions..

- "Good" Boolean functions
- "Good" S-boxes etc…

☺ Provably ensure that many attacks such as differential or linear cryptanalysis will not work.

⛔ Magical objects that make ciphers secure ?



A "Good" Boolean function…

Some Remarks !

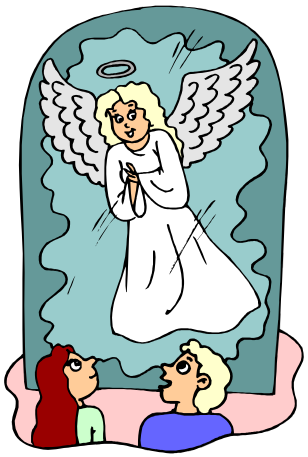
"We can strongly affirm that a very consequent theory of stream encryption exists…"

"Block ciphers are not secure, one should use stream ciphers instead…"

"It is impossible to hide a trapdoor in a stream cipher …"

[Eric Filliol, Plaintext-Dependent Repetition Codes … the AES case, eprint.iacr.org, 2003]

The Tale of "Good" Boolean Functions..

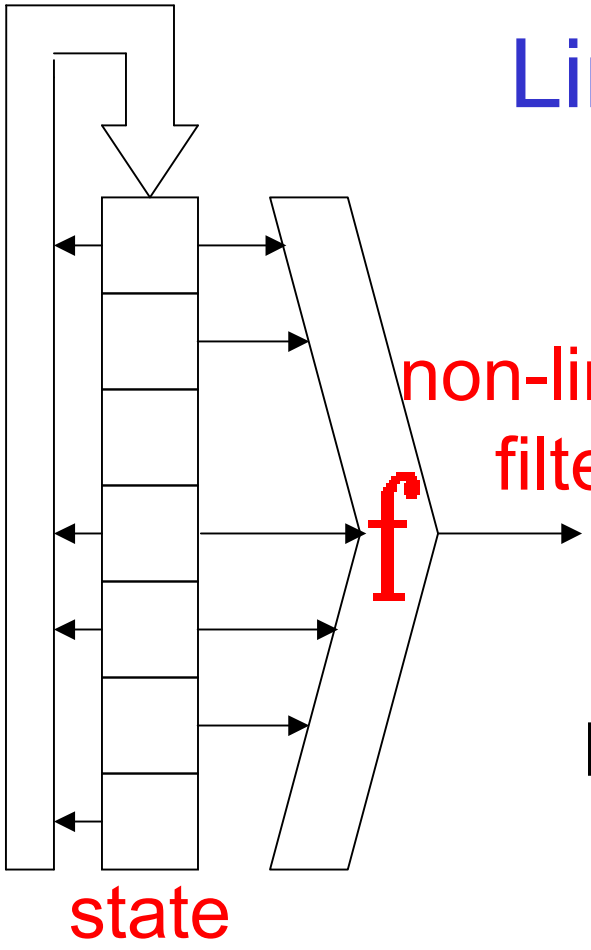


Naïve belief that ciphers build out of such components would be secure.

In fact this approach fails, sometimes quite miserably, to produce secure ciphers:

- Algebraic attacks on AES and Serpent
  [Courtois-Pieprzyk, AsiaCrypt 2002].
- Stream ciphers: much worse, today.
  [For some ciphers, there is no "good" Boolean functions !]

Popular stream ciphers:



Linear sequence generator +

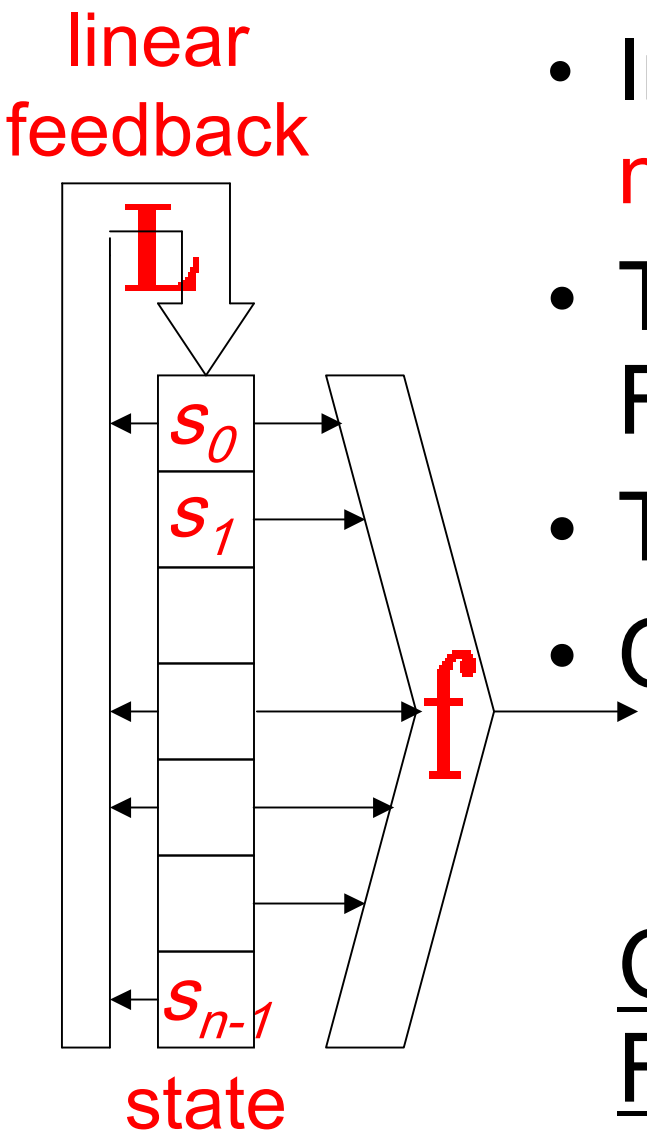
a stateless combiner

$b_0, b_1, b_2, \ldots$

Example: One or several LFSRs

+ a Boolean function.

Notations



- Initial key $k \in GF(2)^n$
  n-bits $k_0, k_1, k_2, \ldots, k_{n-1}$
- The state $s \in GF(2)^n$
  First $s = k$,
- Then $s = L(s)$ etc..
- Output bits: Apply $f(s)$
  $b_i = f(L^i(k))$

Given: some of the b_i
Find: the secret key k

Direct Algebraic Attack Approach:

$$\begin{cases} b_0 = f \quad (k_0, \ldots, k_{n-1}) \\ b_1 = f\,(L\ (k_0, \ldots, k_{n-1})) \\ b_2 = f\,(L^2(k_0, \ldots, k_{n-1})) \\ \vdots \end{cases}$$

Solve this system of equations.

Extremely overdefined even for moderate quantity of keystream, e.g. 20 Kbytes.

Part 1

Algebraic Attacks that are also Higher Degree and Higher Order Correlation Attacks.

cf. Nicolas Courtois: Higher Order Correlation Attacks, XL algorithm and Cryptanalysis of Toyocrypt, ICISC 2002 or eprint.iacr.org

Known Correlation Attacks:

1. Correlation Attacks:
   - exploit a correlation between the output of f and a linear combination of a few inputs.
   - Higher-Order Correlation Attacks: many inputs.

2. Linear Approximation Attacks:
   - maximum order correlation ⇔ a linear approximation of f true with some $P \neq 1/2$.

3. [ICICS'2002]: Higher degree approximations.

⇔ Higher-order correlation attacks with a non-linear function. Apparently not exploited so far…

Another Well Known Attack:

Assumption:

f has a low degree d.

Then: the key is found given $\binom{n}{d}$ keystream bits and within $\binom{n}{d}^{\omega}$ computations,

with ω being the exponent of the Gaussian reduction

$\omega \leq 2.376$ [Coppersmith, Winograd]

- Linearisation: One new variable for each monomial, solve a linear system.

Improvement (1):

What if the amount of the keystream available is not sufficient ?
$\Rightarrow$ use XL/GB instead of linearisation.

The less keystream, the more will be the complexity…

Quickly impractical if $d>6$ or so…

Improvement (2):

Example: Toyocrypt, *n=128, d=63*.

What if the degree *d* is too big ?

Find a low degree approximation !

Low Degree Approximations

We want f to have an approximation of low degree d.

Remark: if such an approximation exist, there are algorithms to find them:

- a.k.a. “Learning Polynomials with Queries in Presence of Noise”
- a.k.a. “Decoding Reed-Muller codes”.

Higher Degree Correlation Attack:

A stream cipher with linear feedback and a Boolean function f. Our assumption: f has a low degree approximation g :

$f(s) = g(s)$ with probability $P = 1 - \varepsilon$

XL will work when:

1. The degree K of $g(s)$ should be low.
   For example will be $d=4$ for Toyocrypt.
2. The ε should be very small.
   For example $\varepsilon = 2^{-17}$ for Toyocrypt.

Toyocrypt

One of the only two stream ciphers accepted to the second phase of CRYPTREC (for the Japanese government).

The design of Toyocrypt

- A bent function [Rothaus recipe]
- add s_{127} to make it balanced.

$$f(s_0,..,s_{127}) = s_{127} + \sum_{i=0}^{62} s_i s_{\alpha_i} + s_{10}s_{23}s_{32}s_{42} + \\ + s_1 s_2 s_9 s_{12} s_{18} s_{20} s_{23} s_{25} s_{26} s_{28} s_{33} s_{38} s_{41} s_{42} s_{51} s_{53} s_{59} + \prod_{i=0}^{62} s_i.$$

with $\{\alpha_0, \ldots, \alpha_{62}\}$ being some permutation of the set $\{63, \ldots, 125\}$.

The case of Toyocrypt

In Toyocrypt, there are only a few higher order monomials.

$$f(s_0,..,s_{127}) = s_{127} + \sum_{i=0}^{62} s_i s_{\alpha_i} + s_{10}s_{23}s_{32}s_{42} + \\ + s_1 s_2 s_9 s_{12} s_{18} s_{20} s_{23} s_{25} s_{26} s_{28} s_{33} s_{38} s_{41} s_{42} s_{51} s_{53} s_{59} + \prod_{i=0}^{62} s_i.$$

An approximation with $d=4$ and $\varepsilon=2^{-17}$.

Solve them – XL/GB Algorithms

- [Shamir, Patarin, Courtois, Klimov, Eurocrypt'2000]
- [Courtois, ICISC'02], [Courtois, Patarin, CT-RSA'03]
- Gröbner bases, Buchberger algorithm, F4, F5, F5/2 by Jean-Charles Faugère… Old papers by Lazard…
- Recent many paper: Claus Diem, Gwenole Ars, Magali Bardet, Jean-Charles Faugère, Bruno Salvy, Makoto Sugita, Mitsuru Kawazoe, Hideki Imai, Jiun-Ming Chen, Nicolas Courtois, Bo-Yin Yang and others.

Paper [Courtois, ICISC 2002]: introduces and analyses a generalized version of XL, for equations of any small degree *d*, not only *2*.

X L means…

- eXtended Linerisation
- Multiply (X) and Linearise
- eXpansion in the ideaL spanned by the equations..
- doing things like x_1 * l_3
- etc…

The principle of XL:

Multiply the initial equations by low-degree monomials:

$$1 = x_5 + x_0x_1 + x_0x_2$$

becomes:

$$x_1 \cdot 1 = x_1 \cdot (x_5 + x_0x_1 + x_0x_2)$$

(degreee 3 now).

The idea of XL:

Multiply equations by low-degree monomials.

- Count new equations: R
- Count new monomials present: T

One term can be obtained in many different ways,

$\Rightarrow$ T grows slower than R.

How XL works:

Initial system: m equations and $\binom{n}{d}$ monomials (terms).

Multiply each equation by a product of any D-d variables so that the total degree is D :

- Equations $R = m \cdot \binom{n}{D-d}$
- Terms $T = \binom{n}{D}$

Idea: One term can be obtained in many different ways, R grows faster than T.

Necessary condition: *R/T > 1*

gives $m \cdot \binom{n}{D-d} / \binom{n}{D} > 1$ and thus $D \approx n/m^{1/d}$

If sufficient, the complexity of XL would be about

$$T^{\omega} \approx \binom{n}{n/m^{1/d}}^{\omega}$$

XL will always work

Remark: !

Over any small finite field, when $D>q$ and the field equations $x_i^q=x_i$ can be included, XL always do work, for ANY SYSTEM OF EQUATIONS (worst case).

[Mireille-Martin Deschamps, private communication]

See: Jacques Patarin and Nicolas Courtois: About the XL algorithm over GF(2), CT-RSA 2003.

XL always works, (here d=2, D=3)

d	2	2	2	2	2	2	2	2	2	2	2	2
n	10	10	10	10	10	20	20	20	20	20	64	64
m	10	14	16	17	18	20	40	50	60	65	512	1024
D	3	3	3	3	3	3	3	3	3	3	3	3
R	110	154	176	187	198	420	840	1050	1260	1365	33280	66560
T	176	176	176	176	176	1351	1351	1351	1351	1351	43745	43745
Free	110	154	174	175	175	420	840	1050	1260	1350	33280	43744

Figure 1: XL simulations for $D = 3$.

n number of variables.

m number of equations.

d the degree of the initial equations to be solved.

D we generate equations of total degree $\leq D$ in the x_i.

R number of equations generated (independent or not). $R = m \cdot \binom{n}{D-2}$

T number of monomials of degree $\leq D$ $T = \binom{n}{D}$

Free number of linearly independent equations among the R equations.

◇ **XL will work when** $Free \geq T - D$.

XL always works… (Examples with D>2)

K	3	3	3	3	3	3
n	10	10	10	10	10	10
m	10	10	10	10	10	10
D	3	4	5	6	7	8
R	10	110	560	1760	3860	6380
T	176	386	638	848	968	1013
$Free$	10	110	560	846	966	1011

3	3	3	3	3
16	16	16	16	16
16	16	16	16	16
3	4	5	6	7
16	272	2192	11152	40272
697	2517	6885	14893	26333
16	272	2192	11016	26330

K	4	4	4	4	4	4	4
n	10	10	10	10	10	10	10
m	10	10	10	10	10	10	10
D	4	5	6	7	8	9	10
R	10	110	560	1760	3860	6380	8480
T	386	638	848	968	1013	1023	1024
$Free$	10	110	560	966	1011	1021	1022

4	4	4	4	4
16	16	16	16	16
16	16	16	16	16
4	5	6	7	8
16	272	2192	11152	40272
2517	6885	14893	26333	39202
16	272	2192	11152	39200

The behaviour of XL (d=2)

It is possible to predict the exact number of linearly independent equations in XL.

D	$Free$
3	$Min\left(T, R\right)$
4	$Min\left(T, R - \binom{m}{2} - m\right)$
5	$Min\left(T, R - (n+1)\binom{m}{2} - (n+1)m\right)$
6	$Min\left(T, R - \left[\binom{n}{2} + \binom{n}{1} + \binom{n}{0}\right] \cdot \left[\binom{m}{2} + \binom{m}{1}\right] + \binom{m}{3} + m^2\right)$

The behaviour of XL (d>2)

We can also predict it, for all useful values (here done “by hand”, general case: see recent papers by Faugère, Diem, Bo-Yin Yang etc.).

D	$Free$
$K..2K-1$	$Min(T,R)$
$2K..3K-1$	$\min\left(T, R-\left(\sum_{i=0}^{D-2K}\binom{n}{i}\right)\left(\binom{m}{2}+m\right)\right)$
$3K..4K-1$	$\ldots$

XL Attack on Toyocrypt

Solve:

$$\begin{cases} b_0 = f \quad (k_0, \ldots, k_{n-1}) \\ b_1 = f\left(L\ (k_0, \ldots, k_{n-1})\right) \\ b_2 = f\left(L^2(k_0, \ldots, k_{n-1})\right) \\ \quad \vdots \end{cases}$$

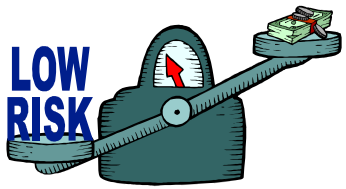
Write instead an approximation of degree $d=4$. Since $\varepsilon=2^{-17}$ we may take about 2^{17} approximations that will be simultaneously true !

XL Attack on Toyocrypt

Given any 2^{17} keystream bits, do not have to be consecutive, we get a system to solve with XL, $D=9$.

A direct Application of XL gives:

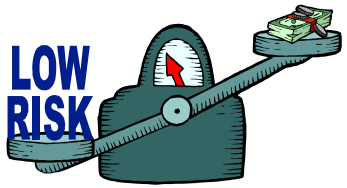
$$2^{122}$$



A trade-off

Take more equations, a system is more overdefined, and then the complexity of XL will substantially decrease, because a lower D is sufficient, then repeat the whole.

Exploring the trade-off:
For $D=6$ we get 2^{92}



So far..

2^{92} ?

There are much better attacks

[Mihaljevic, Imai, IEICE 2002]

However,

- XL can break other stream ciphers that resist to all known attacks.
- Mihaljevic and Imai need 32 consecutive keystream bits, XL attacks doesn't.
- Ciphertext-only attacks are possible: e.g. ASCII English - every 8^{th} bit is 0.

Part 2

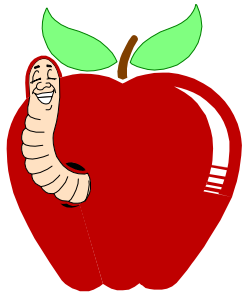
Attacks Using Implicit Equations

Cf. Nicolas Courtois, Willi Meier: Algebraic Attacks on Stream Ciphers with Linear Feedback, Eurocrypt 2003.

Additional Weakness of Toyocrypt

$$f(s_0,..,s_{127}) = s_{127} + \sum_{i=0}^{62} s_i s_{\alpha_i} + s_{10}s_{23}s_{32}s_{42} + \\ + s_1 s_2 s_9 s_{12} s_{18} s_{20} s_{23} s_{25} s_{26} s_{28} s_{33} s_{38} s_{41} s_{42} s_{51} s_{53} s_{59} + \prod_{i=0}^{62} s_i .$$

The parts of degree *4, 17* and *63* are divisible by a common factor: $s_{23}\, s_{42}$.



$f(s)=1 \Rightarrow f(s)\,(s_{23}+1) = (s_{23}+1)$.

An equation of degree d=3,

true with probability 1.

New Type of Attack

By multiplying the equations by a well chosen polynomial, their degree can be reduced from $d=4$ to 3.

Moreover we reduce ε from 2^{-17} to 0.

New Type of Attack

This is no longer a higher-order correlation attack !

A purely algebraic attack.

New attack and XL:

We do not need XL anymore:

- For equations of degree d true with probability 1,

$$\Rightarrow \text{take } \binom{n}{d} \text{ equations,}$$

old linearization / new equations. $\mathbf{WF} = \binom{n}{d}^{\omega}$

- However, if $\binom{n}{d}$ keystream bits are not available, use XL, [cf. extended version.]

! Much Lower Complexity:

For $n=128$, $d=3$, $\varepsilon=0$.

- Given 2^{18} keystream bits - $\frac{1}{2}\binom{n}{3}$
- Using 2^{37} bits of memory - $\binom{n}{3}^2$
- The secret key can be recovered in 2^{49} CPU clocks - $\binom{n}{3}^{\omega}$.
- Verified experimentally.

Toyocrypt is very weak

The cipher that could have become a standard for the Japanese government…,

can be broken in

2^{49} CPU clocks.

(few days on a PC)

What's New ?

Solve equations $f(s)=1$ with $s = L^t(k)$.

S1. [folklore] If f is of low degree d, solve by linearization attack in $\binom{n}{d}^{\omega}$.

S2. [Courtois, ICICS'02]: f is approximated by a function of low degree d, solve by XL.

S3. Now: [EC'03]: Given f, there is a Boolean function g such that $f(s) * g(s)$ is of low degree.

S4. Also: As 3. but true with some probability…

Overview of the New Attack

Instead of

$f(s) = b_t$ with $s = L^t(k)$,
solve the equations

$f(s) * g(s) = b_t * g(s)$ with a well chosen g.

N.B. This is <u>not yet</u> the most general algebraic attack on stream ciphers, see the papers…

Q: Do such $g(s)$ exist ?

Problem:

Find g such that

$$f(s) * g(s) = b_t * g(s)$$

is of low degree.

More General Attack S5:

[Courtois-Meier, Krause-Armknecht]:

Combine several consecutive (or not) equations:

$$\begin{cases} f(k) = 1 \\ f'(k) = 0 \\ f''(k) = 1 \\ \vdots \end{cases}$$

$$f(k)\cdot g(k)+f'(k)\cdot g'(k)+f''(k)\cdot g''(k) = g(k)\cdot 1+g''(k)\cdot 2$$

Also works for combiners with memory bits, they get eliminated !

General Algebraic Attack (S5)

Problem:

Find functions *g(s), g'(s), …* such that

$$f(k)\cdot g(k)+f'(k)\cdot g'(k)+f''(k)\cdot g''(k)=g(k)\cdot 1+g''(k)\cdot 2$$

becomes of low degree.

Do such *g(s), g'(s), …* exist ?

New design criteria on all LFSR-based regularly clocked stream ciphers.

Turns out to be almost identical to both:

- the security criterion defined in Section 2 of "Cryptanalysis of HFE", N. Courtois, CT-RSA 2001.
- the requirements advocated by Courtois and Pieprzyk for the S-boxes of block ciphers.

In some cases, they ALWAYS do exist !

A Boolean function +
a small number of inputs,

[say $k=10$ as in LILI-128]

$\Rightarrow$ such g always do exist.

$f(s) * g(s) = b_t * g(s)$ of low degree ?

Small number of inputs,

[say $k=10$ as in LILI-128]

$\Rightarrow$ such g always do exist.

Finding the *g(s)*

Consider the following multi-sets:

$$A = \{f(x), f(x) \cdot x_1, f(x) \cdot x_2, \ldots, f(x) \cdot x_{k-1}x_k, \ldots\}$$

$$B = \{1, x_1, x_2, \ldots, x_{k-1}x_k, \ldots\}$$

$$C = A \cup B$$

Basic idea:

dimension $\leq 2^k$.

If sufficiently many polynomials in C,

then linear dependencies will exist !

First try:

$$A = \{f(x), f(x) \cdot x_1, f(x) \cdot x_2, \ldots, f(x) \cdot x_{k-1}x_k, \ldots\}$$

Linear dependencies in A:

Each gives a *g* such that *f*g = 0*.

In order to have |A| = 2^k polynomials,

we need to consider

g of degree up to *k*.

Not very interesting !

Better version:

$$A = \{f(x), f(x) \cdot x_1, f(x) \cdot x_2, \ldots, f(x) \cdot x_{k-1}x_k, \ldots\}$$

$$B = \{1, x_1, x_2, \ldots, x_{k-1}x_k, \ldots\}$$

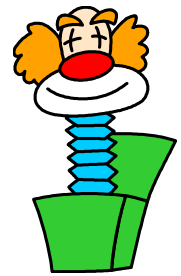
$$C = A \cup B$$

Linear dependencies in A ∪ B : Gives *g* such that *f*g = h* is of small degree.

We only need $|A| = 2^{k-1}$ and $|B| = 2^{k-1}$.

Great improvement ? Yes !

Is 2^{k-1} much smaller than 2^k ?



Here, yes !!!

$$A = \{f(x), f(x) \cdot x_1, f(x) \cdot x_2, \ldots, f(x) \cdot x_{k-1}x_k, \ldots\}$$

- What is the degree (of g)to be used to have $|A| = 2^k$?

$$\Longrightarrow \text{d=k, because } \sum_{i=0}^{k} \binom{k}{i} = 2^k.$$

- What is the degree (of g) to be used to have $|A| = 2^{k-1}$?

$$\Longrightarrow \text{d=k/2, because } \sum_{i=0}^{k/2} \binom{k}{i} \approx 2^{k-1}.$$

Theorem:

For ANY Boolean function $f(x)$

using k variables,

there is a Boolean function

$g(x) \neq 0$ of degree at most $k/2$

such that:

$f(x)*g(x) = h(x)$

is of degree at most $k/2$.

Theorem:

Any Boolean function of using *k*

Variables leads to relations of degree at most *k/2*.

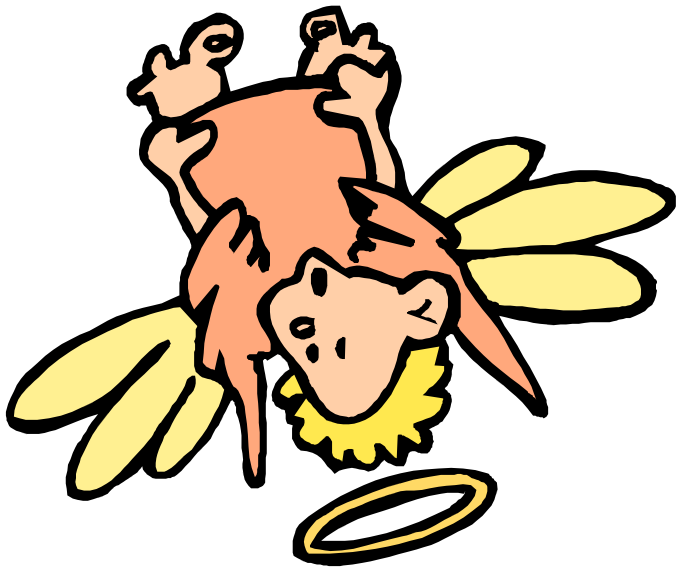
The "Tale of

Good Boolean Functions"

fails: Some ciphers

are broken for

any Boolean function.



Part 2.1.

Application to LILI

[still Eurocrypt 2003]

LILI-128

One of the NESSIE candidates,
claimed very secure,
rejected

(all the other stream ciphers were rejected too !)

Second Component of LILI-128

Not very strong, *k=10*,
equations of degree *d=5* can be generated
whatever is the Boolean function used,
(worst case).
$\Rightarrow$ Attack in $\binom{n}{5}^{\omega}$.

Simulations show that,
for the specific function *f* used, *14*
equations of degree *d=4* can be generated.
Bad choice !
$\Rightarrow$ Attack in $\binom{n}{4}^{\omega}$.

The whole LILI-128

This attacks can be extended also for stream ciphers that are not regularly clocked:

Example:

For LILI-128 we can:

- Guess the state of the first LFSR $\Rightarrow$ 39 bits.
  Works because the algebraic attacks works given any set of keystream bits at known positions, not only if consecutive.
- Clock the clocking component around $\Rightarrow 2^{39}-1$ clocks at a time.
- If the first component is an LFSR, $\Rightarrow$ combination of both, decimation attack [Filliol].

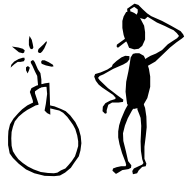
Example of an Attack on LILI-128

- First LFSR: $2^{39}-1$ clocks at a time.
- Second LFSR: $n=89$.
- Our equations: $d=4$.
- Given 2^{57} keystream bits - $2^{39} \cdot \binom{n}{4}^{\omega}$
- Using 2^{43} bits of memory -$\binom{n}{4}^{2}$
- The secret key can be recovered in 2^{57} CPU clocks - $\binom{n}{4}^{\omega}$.

The Power of the New Attack

- Works on any regularly clocked stream cipher with linear feedback and a non-linear filter, and for any Boolean function such that:
  - Either uses a small subset of state bits…
  - Or is very (rather extremely) sparse…
  - Or can be factored with a low degree factor…
  - Or can be approximated by one of the above…
  - Or it's part of high degree is one of the above…
- For specific ciphers works even better than the worse case (random Boolean function).

Asymptotic and Concrete Aspects:

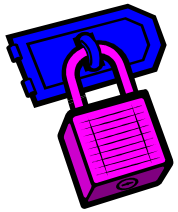
- We can break any stream cipher with linear feedback and any Boolean function on a small subset of k state bits in $\binom{n}{k/2}^{\omega}$. !
- Polynomial if k is seen as a small constant.
- Surprising ?  No. with linearisation, any such cipher is known to broken just in $\binom{n}{k}^{\omega}$.

Already polynomial, However:

- In concrete terms, the new attack complexity is in roughly the square root of the old one. !

Concrete Aspects:

- Old (generic) attack: $\binom{n}{k}^{\omega}$.
- Thus many stream ciphers, frequently unpublished and proprietary, have been designed so that, one has for example $\binom{n}{k}^{\omega} = 2^{80}$



- New generic attack: $\binom{n}{k/2}^{\omega} \approx$ the square root.

- Now we can break all these ciphers in roughly 2^{40} !
- The examples of Toyocrypt and LILI-128 show that for specific ciphers, the resistance against algebraic attacks may be substantially worse…

***What's New Here

Relation with the linear complexity [lc].

1. Known Before:

- For a combiner with k inputs, the linear complexity is at most $\binom{n}{k}$.
- From Sean Murphy's crypto course [RHUL, Stream-II, p. 9, 2003]:
  - If a sequence $s_0,\ldots,s_{n-1}$ is used as an enciphering sequence, its (local) linear complexity should be $> 2^{k/2}$, where 2^k is the key diversity (otherwise there is an attack better than exhaustive key search)

$\Rightarrow$ Linear Complexity should be $lc > 2^{40}$ to avoid an attack in 2^{80}. Our attack is in $\binom{n}{k/2}^{\omega} \approx \binom{n}{k}^{\omega/2}$.
So we need about $lc > 2^{60}$.

$\Rightarrow$ Systems designed in the past are likely to be broken.

Diversity of Algebraic Attacks

May work for many reasons
and be based on
very different techniques
to find equations:

- Factoring multivariate polynomials [Toyocrypt].
- Finding implicit equations by Gaussian reduction [LILI-128].
- Clever equations found by hand [Armknecht E0].
- New very fast technique: [Crypto 2003].

Extensions of the New Attack

- Extension to stream ciphers that are not regularly clocked: Attack on LILI-128 in 2^{57}.

- Can be extended to stream ciphers that are using a stateful combiner, instead of a (stateless) Boolean function.

  – Demonstrated first by Frederik Armknecht, EO generator, 2^{70}. [eprint 12/2002, Crypto 2003]

  – Improved by Courtois to 2^{49} [Crypto 2003].

Part 3

From Algebraic Attacks

to…

Part 3

Fast Algebraic Attacks

Cf. Nicolas Courtois: Fast Algebraic Attacks on Stream Ciphers with Linear Feedback, Crypto 2003.

Can We Do Better ?

If the keystream bits are consecutive;
Yes, much better !

Nicolas Courtois: "Fast Algebraic Attacks on Stream Ciphers with Linear Feedback".
Crypto 2003.

What Are Fast Algebraic Attacks ?

$$S3 \rightarrow S5$$

Use some equations of type S3 and transform into equations of type S5 by a pre-computation !

How it is done ?

"Solve" the same S3 equations…

Complexity = T^{ω}

- Naïve method: T^{3}
- Strassen: $T^{\log_2(7)}$
- Coppersmith: $T^{2.376..}$
- If it were sparse: T^{2}
- Trick: nearly T^{1} ?

Gaussian Reduction in Linear Time ?

These equations are very special:

One equation,

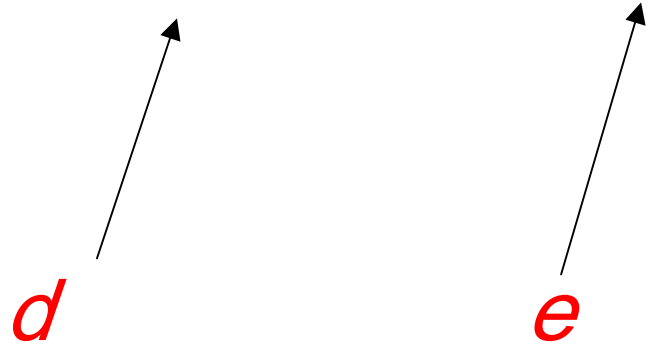
re-applied to any state [window] !

$f(s) * g(s) = b_t * g(s)$ with $s = L^t(k)$.

changes

Gauss in Linear Time ???

$f(s) * g(s) = b_t * g(s)$ with $s = L^t(k)$.



degree

Idea 1: Usually $d>e$

LILI-128: 4>2, Toyocrypt 3>1, …

Gauss in Linear Time ???

$f(s) * g(s) = b_t * g(s)$ with $s = L^t(k)$.

d e

Idea 3: If we have $\binom{n}{d}$ keystream bits, eliminate the left sides and we get equations of degree only e.

What for ?

Idea 4. Pre-computation Attack

[first suggested to me by Philip Hawkes]

Eliminate the left sides and we get equations of degree only e.

Idea 5: Left sides do not depend on b_t.

Can be done once for all !

- Pre-computation – $\binom{n}{d}^{\omega}$
- To break cipher given output bits - $\binom{n}{e}^{\omega}$.

So What ?

- Old Attack – $\binom{n}{d}^{\omega}$
- New Attack – $\binom{n}{e}^{\omega}$
- Pre-computation - $\binom{n}{d}^{\omega}$.

Nicer but not fundamentally faster…

Gauss in Linear Time ?

Idea 6:

- Up till now – all algebraic attacks worked given (almost) any subset of keystream bits.
- What if the keystream is from consecutive states ?

Gauss in Linear Time ?

Idea 6 contd. Consecutive states ?

Recursive structure in left sides !

$$\begin{cases} fg & (k_0, \ldots, k_{n-1}) \\ fg & (L(k_0, \ldots, k_{n-1})) \\ fg & (L^2(k_0, \ldots, k_{n-1})) \\ & \vdots \end{cases}$$

Gauss in Linear Time ?

- Compute these equations: $\binom{n}{d}^2$
- Compute a linear dependency:
  only about $\binom{n}{d} \log \binom{n}{d}$

$$\begin{cases} fg \quad (k_0, \ldots, k_{n-1}) \\ fg\,(L\,(k_0, \ldots, k_{n-1})) \\ fg\,(L^2(k_0, \ldots, k_{n-1})) \\ \vdots \end{cases}$$

Gauss in Linear Time ?

Idea 7: There are linear dependencies
that DO NOT depend on the key !

Theorem: When a key is fixed,
and for any key, the sequence will be
produced by a large LFSR
of length $\leq \binom{n}{d}$.

Reason: recursive structure…

Gauss in Linear Time ?

Idea 8. Instead of working on large equations, work on binary sequences, size $\approx \binom{n}{d}$.

1. Compute the sequence.
2. Find the linear recurrence by Berlekamp-Massey Algorithm.

Two Miracles.

Idea 9. (or Miracle 1).

1. Compute the sequence.
   Will take only $\mathbf{n} \cdot \binom{\mathbf{n}}{\mathbf{d}}$ operations.

- 1.a. $\mathbf{k'}, \mathbf{L(k')}, \mathbf{L^2(k')}, \mathbf{L^3(k')}, \ldots$

(advance LFSR takes $\leq \mathbf{n}$ operations).

- 1.b. Apply $\mathbf{f} \cdot \mathbf{g}$: constant or small.

Not in general, just for every stream cipher I've ever seen !

Two Miracles…

Idea 10. (or Miracle 2).

2. Compute the recurrence

It will take:

- Berlekamp-Massey: $\binom{n}{d}^2$
- Fast versions: about $\binom{n}{d} \log \binom{n}{d}$
  - R. E. Blahut: Theory and Practice of Error Control Codes, Addison-Wesley, 1983.
  - Jean-Louis Dornstetter: On the Equivalence Between Berlekamp's and Euclid's Algorithms.
  - R. P. Brent, F. G. Gustavson, and D. Y. Y. Yun: Fast solution of Toeplitz systems of equations and computation of Padé approximants.

Gauss in Linear Time !

All steps are now done in time essentially linear in $\binom{n}{d}$!

(about $\binom{n}{d}(n + d \log n)$).

Two Views

1. Solve the old equations in a new way.
2. Use the old S3 equations to + Berlekamp-Massey to derive new S5 equations…

These new equations use GREAT MANY output bits.

"Wide" Equations..

These new equations use GREAT MANY output bits.

- How to make sure that do not exist ???
- Impossible, found in "almost linear" time.
- The left sides: sort of shortcut:
  - Final right side combinations can be found in time very roughly about $\binom{n}{d}^{\omega}$.
  - Given the left sides, only "essentially" $\binom{n}{d}$.

Scary "Wide" Equations..

Goal: design an LFSR-based stream cipher with security 2^{128}.

Problem: How to make sure that there is no algebraic relation of size 2^{100} that relates key bits and output bits ?

Example: Linear complexity may be 2^{100}.
I cannot check if exists.

Scary Algebraic Equations..

Problem: How to make sure that there is no algebraic relation of size 2^{100} ?

Crypto'03 paper clearly demonstrates that in MANY interesting cases you cannot be sure unless you can do about 2^{100} computations.

Also works for linear complexity (a class of ciphers that can be broken in a time $\approx$ linear complexity).
Murphy's course: should be 2^{40}. Not enough !!!
Many other relations may exist…

**Concrete Results

Gives the best attack known so far for 4 well known stream ciphers:

- Toyocrypt – Cryptrec submission $\approx 2^{25}$
- LILI-128 – Nessie submission $\approx 2^{31}$
- E0 – Bluetooth keystream generator $\approx 2^{49}$
- SFINKS – Estream submission [2005] $\approx 2^{70}$

**Results vs. Previous Attacks

cryptosystem	Toyocrypt			LILI-128				EO	
n	128	128	128	89	89	89	89	128	128
d		3	3	4	4		4	4	4
e			1				2		2
Paper	Imai	EC'03	new	EC'03	EC'03	077	new	Armknecht	new
Data	2^{48}	2^{18}	2^{18}	2^{18}	2^{57}	2^{46}	2^{60}	2^{24}	2^{24}
Memory	2^{48}	2^{37}	2^{14}	2^{43}	2^{43}	2^{51}	2^{24}	2^{48}	2^{37}
Pre-computation	2^{80}		$O(2^{23})$			2^{56}	$O(2^{26})$		$O(2^{28})$
Attack Cxty	2^{64}	2^{49}	2^{20}	2^{96}	2^{57}	2^{56}	2^{31}	2^{64}	2^{49}

Follow-up work…

Papers/Results:

1. Nicolas Courtois: Higher Order Correlation Attacks, XL algorithm and Cryptanalysis of Toyocrypt, ICISC 2002 or eprint.iacr.org
   ⇒ 2^{92} for Toyocrypt.
2. Nicolas Courtois, Willi Meier: Algebraic Attacks on Stream Ciphers with Linear Feedback, EuroCrypt 2003.
   ⇒ 2^{49} for Toyocrypt, 2^{57} for LILI-128.
3. Nicolas Courtois: Fast Algebraic Attacks on Stream Ciphers with Linear Feedback, CRYPTO 2003.
   ⇒ $O(2^{23})$ for Toyocrypt, $O(2^{31})$ for LILI-128, $O(2^{49})$ for E0.

Follow-Up Papers:

4. Frederik Armknecht: Improving Fast Algebraic Attacks, FSE 2004.
5. Philip Hawkes and Gregory G. Rose: Rewriting Variables: the Complexity of Fast Algebraic Attacks on Stream Ciphers, Crypto 2004.

Initial claims by Courtois:

$\Rightarrow$ $O(2^{23})$ for Toyocrypt, $O(2^{31})$ for LILI-128, $O(2^{49})$ for E0.

Best Rose-Hawkes results with FFT (add log):

$\Rightarrow$ $O(2^{30})$ for Toyocrypt, $O(2^{40})$ for LILI-128, $O(2^{49})$ for E0.

Broken at the First Glance…

In 2005 Braeken, Lano, Mentens, Preneel and Varbauwhede have invented a new stream cipher:

- SFINKS – ECRYPT submission $\approx 2^{70}$

6. Nicolas Courtois: Cryptanalysis of Sfinks. ICISC'05 and eprint/2005/243

Simply broken once you take the time to examine the (already known) algebraic attack – BUT need to handle many computer simulations to determine if there exist suitable equations, no theoretical method to predict the result...

Simulations on SFINKS

☺ Automatic Cryptanalysis ☺

degree e of g	0	1	2	2	3	3	3	4	4	4	5	5
degree d of $h = fg$	15	14	7	8	6	7	8	5	6	7	5	6
dimension A of these g	1	0	0	6	0	32	136	0	4	204	0	4
out of which we used	–	–	–	1	–	1	1	–	1	1	–	1

complexity according to Hawkes-Rose FFT estimations												
pre-computation step	–	–	–	$2^{59.8}$	–	$2^{54.5}$	$2^{59.8}$	–	$2^{49.0}$	$2^{54.5}$	–	$2^{49.0}$
substitution step	–	–	–	$2^{69.7}$	–	2^{71}	2^{76}	–	$2^{71.6}$	$2^{76.9}$	–	$2^{77.3}$
solving step	–	–	–	$2^{42.1}$	–	$2^{60.1}$	$2^{60.1}$	–	$2^{76.9}$	$2^{76.9}$	–	$2^{92.8}$
keystream required	–	–	–	$2^{48.6}$	–	$2^{43.6}$	$2^{48.5}$	–	$2^{38.5}$	$2^{43.6}$	–	$2^{38.5}$

Problem with Fast Algebraic Attacks

Very “wide” equations”.

- How to make sure there is no such equations ? Hard.
- [If exist, how to find them efficiently ? – shortcuts using the special structure of the cipher cannot be excluded !]

Need for provable security against [fast] algebraic attacks… Open.

Part 4

Algebraic Attacks, Boolean Functions Annihilators and More (!)

Paradigm Shift

[Shannon, Jakobsen, Patarin, Pieprzyk, Courtois et al]

Do not only look at functions, look at multivariate algebraic relations (implicit equations).

Claim: This is the most general formulation of algebraic attacks [Carlet's Algebraic Immunity is only a special case].

Unified view of Algebraic Attacks

Non-existence of small multivariate relations between inputs/outputs.

- Applies to multivariate public key cryptosystems: Sflash, Quartz
- Applies to the non-linear part of a stream cipher, even if stateful.
- Applies to the S-boxes of a block cipher

Nicolas Courtois: General Principles of Algebraic Attacks and New Design Criteria for Components of Symmetric Ciphers, Invited talk, AES 4 Conference, LNCS 3373, Springer.

Def: "I / O Degree" = "Graph Al"

Consider function $f : GF(2)^n \rightarrow GF(2)^m$,
$f(x) = y$, with $x = (x_0, \ldots, x_{n-1})$, $y = (y_0, \ldots, y_{m-1})$.

Definition [The I/O degree] The I/O degree of f is the smallest degree of the algebraic relation

$$g(x_0, \ldots, x_{n-1}; y_0, \ldots, y_{m-1}) = 0$$

that holds with certainty for every couple (x, y) such that $y = f(x)$.

Application to Stream Ciphers

Nicolas Courtois, Willi Meier: Algebraic Attacks on Stream Ciphers with Linear Feedback, Eurocrypt 2003.

Will Meier, Enes Pasalic and Claude Carlet: Algebraic Attacks and Decomposition of Boolean Functions, Eurocrypt 2004.

=> RE-FORMULATION,
MUCH LESS GENERAL!!!!

Carlet et al. Approach:

Lost of generality/power in four important directions:

- Over GF(2) only
- 1 output Boolean function only
- Probability 1 only
- Can be specifically chosen to eliminate some variables.

The original formulation is infinitely more powerful.

Later attempts to generalize Carlet's approach to these 4 cases are not successful (and cannot be).
For example they work in theory by fragmenting one single algebraic attack that works into a very large number of attacks that works with negligible probability – i.e. this doesn't work in practice…

Basic Problem:

Find *g,* such that

$$f(s) * g(s) = h(s)$$

is of low degree.

(with probability $1-\varepsilon$)

BTW, Remark (for Mathematicians):

Somewhat similar to Padé approximants

$$f(s) * g(s) = h(s)$$

is of low degree

with probability $1-\varepsilon$

More Follow-Up Papers:

7. Will Meier, Enes Pasalic and Claude Carlet: Algebraic Attacks and Decomposition of Boolean Functions, Eurocrypt 2004.
8. Frederik Armknecht: On the Existence of low-degree Equations for Algebraic Attacks, SASC 2004 et eprint.iacr.org/2004/185/ (nothing new).
9. Jovan Dj. Golic: Vectorial Boolean functions and induced algebraic equations, eprint.iacr.org/2004/225/.

Basic Problem:

Find appropriate g, and use

$$f(s) * g(s) = b_t * g(s)$$

being of low degree

Initial Classification

[Courtois-Meier EC'03]

Attack scenario considered	Degree of f	Degree of g	Degree of fg	Use the equation	Only when	Number of equations for m keystream bits
S3a	high	low, $g \neq 0$	low, $fg \neq 0$	$f(s) \cdot g(s) = b_t \cdot g(s)$	always	m
S3b	high	low, $g \neq 0$	$fg = 0$	$g(s) = 0$	$b_t \neq 0$	$m/2$
S3c	high	high	low, $fg \neq 0$	$f(s) \cdot g(s) = 0$	$b_t = 0$	$m/2$

Table 1: Different methods to obtain low degree equations from keystream bits

Later Classification [Carlet..]

In fact first introduced 1 year earlier by [Courtois-Meier, extended version of EC'03, Appendix B] to solve the problem of linear dependencies, proposed under a different name by [Carlet-Meier-Pasalic EC'04] and used by everybody since…

Attack scenario considered	Degree of f	Degree of g	Use g such that	Use the equation	Only when	Number of equations for m keystream bits
$S3_0$	high	low, $g \neq 0$	$f(s)=0 \Rightarrow g(s)=0$	$g(s)=0$	$b_t=0$	$m/2$
$S3_1$	high	low, $g \neq 0$	$f(s)=1 \Rightarrow g(s)=0$	$g(s)=0$	$b_t=1$	$m/2$

Table 1: The alternative classification of ways to obtain low degree equations from keystream bits

These g are annihilators for f and f+1 respectively (!)

Annihilators and Co.

Carlet, Meier and Pasalic “renamed”
the scenarios $S3_x$ as
“finding annihilators to f+x”. (the same thing).

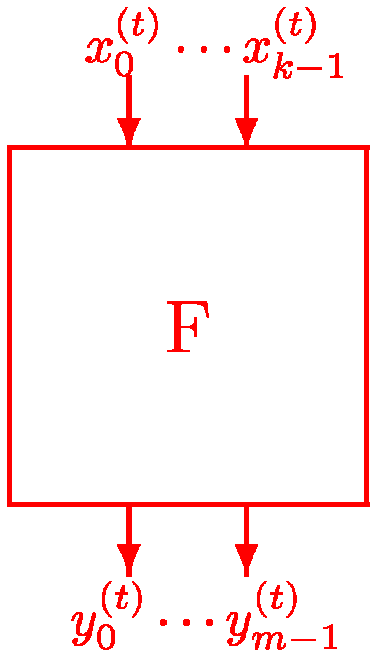
Remark:

Annihilators for f ⇔ Absorbing elements for f+1.

f * g = 0 ⇔ (f +1) * g = g (!)

Two approaches to the same attack…

Each is as good ! – Look at the general case – several outputs.



1. The extended annihilator approach – fixed values for all, look at algebraic equations true with probability 1 on inputs.
2. Original algebraic relation approach: not less interesting.

None is better…

1. Extended annihilator approach – fix output.
2. The initial algebraic relation approach.

1 is better because…

- Less monomials.
- For some output vectors, the degree may be much lower.
- In some cases only some selected output bits appear in the equation, then it does not make sense to write many monomials that will be unused.

2 is better because…

- Uniform approach, can eliminate b_t later, only with this fast algebraic attacks are possible !
- We can use some inputs / special linear combinations of inputs….
- In some cases the attack may use so many outputs, that we cannot fix them (event happens with a negligible probability) !

One output m=1, strange scenario S3c

$$f(s) * g(s) = h(s)$$

- h of low degree,
- g of high degree ?

  still usable when $b_t = 0$:
- just use $h(s) = 0$, only when $b_t = 0$.

[Carlet, Meier, Pasalic]:

$$f * h = f * f * g = f * g = h$$

Back in scenario S3a !

S3c is redundant…

Really ? S3c over GF(q), q>2.

$$f(s) * g(s) = h(s)$$

- h of low degree,
- g of high degree,
- use $h(s) = 0$, when $b_t = 0$.

Try again: $f * h = f * f * g = ?$ … fails.

=> General case q>2: the scenario S3c makes sense.

AND we are still using a function h,
such that $f(s) = 0$ => $h(s) = 0$.

=> The scenario $S3_0$ [Courtois-Meier].

=> [Carlet,…]: we have no annihilator here !

[Nb. Armknecht will find one for some char. function…]

Summary:

GF(q), q=2

- Annihilators/absorbers are everywhere [Carlet et al.], is an equivalent formulation of $S3_0$, $S3_1$ by [Courtois-Meier].

GF(q), q>2

- Use scenarios S3a, S3b, S3c [Courtois-Meier] – known to be redundant – give dependent algebraic equations.
- Better method: scenarios $S3_0$, $S3_1$, … , $S3_{q-1}$ in the original formulation of [Courtois-Meier] !

What about vectorial case - several outputs ?

Extending $S3_0$ to several outputs.

$S3_0$: Find a low degree function *h* such that

$$f(s) = 0 \Rightarrow h(s) = 0 .$$

For vectorial functions m>1, we have related to the notion of affine multiples:

- [Blake, Vanstone, et al: "Applications of Finite Fields" p.25, 1993].
- [Patarin, ext. ver. HFE paper, 1996, p. 15].

$f(s) = t \Rightarrow A(s,t) = 0$; linear in the s_i…

More generally, we need:

"low degree multiples" !

$f(s) = t \Rightarrow A(s,t) = 0$; low degree in s_i…

Low Degree Multiples – not new !

[Patarin Crypto'95]: there are affine multiples for Matsumoto-Imai cryptosystem.

[Patarin Eurocrypt'96, simulations made by Courtois]: there are **no** affine multiples for HFE.

[Courtois RSA-CT'01, Faugère-Joux Crypto 2004]: there are low degree multiples for HFE.

Low Degree Multiples…

By the way, are the "Low Degree Multiples" really Multiples ? Not obvious from def…

$f(s)-t=0$ => $A(s,t) = 0$; low degree in s_i…

Thm. 1. The set of such A is an ideal.

Thm. 2. It is of the form $g^*(f(s) - t)$.

One output, no memory: shown in [Carlet…]. See also [Batten, Indocrypt'03]

Low Degree Multiples…

Finally – equivalent version or another wording of it [Courtois-Pieprzyk version]:

Algebraic Relations paradigm:

$$A(s,t) = 0$$

(can denote really the same thing that Low Degree Multiples)
and can be still more general – eliminate extra variables (!)

Another Paper:

Algebraic Relations paradigm:

$$A(s,t) = 0$$

Applications: block ciphers, stream ciphers, PK crypto. See:

10. Nicolas Courtois: General Principles of Algebraic Attacks and New Design Criteria for Components of Symmetric Ciphers, Survey, block+stream ciphers+HFE. Invited talk, AES 4 Conference, to appear in LNCS 3373, Springer.

I / O Degree of Algebraic Relations

Consider function $f : GF(2)^n \rightarrow GF(2)^m$,
$f(x) = y$, with $x = (x_0, \ldots, x_{n-1})$, $y = (y_0, \ldots, y_{m-1})$.

Definition [The I/O degree] The I/O degree is the smallest degree of the algebraic relation

$$g(x_0, \ldots, x_{n-1}; y_0, \ldots, y_{m-1})$$

that holds with certainty for every couple (x, y) such that $y = f(x)$.

A "good" cipher should use at least some components with high I/O degree.

Claim / Proposal

This should be necessary for the security of:

- S-boxes in Block Ciphers
- Combiners in Stream Ciphers
- Trapdoor Functions (PK crypto).

(according to Courtois, Pieprzyk, Jakobsen, Patarin, Shannon etc…)

Yet another paper:

11. Nicolas Courtois: Algebraic Attacks on Combiners with Memory and Several Outputs, eprint.iacr.org/2003/125/ (updated recently) and ICISC 2004.

Part 5

Attacking Any* Combiner/Filter

(General Case with Many Outputs)

cf. Nicolas Courtois: Algebraic Attacks on Combiners with Memory and Several Outputs, eprint.iacr.org/2003/125/ and ICISC 2004.

* worse case attacks that break any component of the same size.

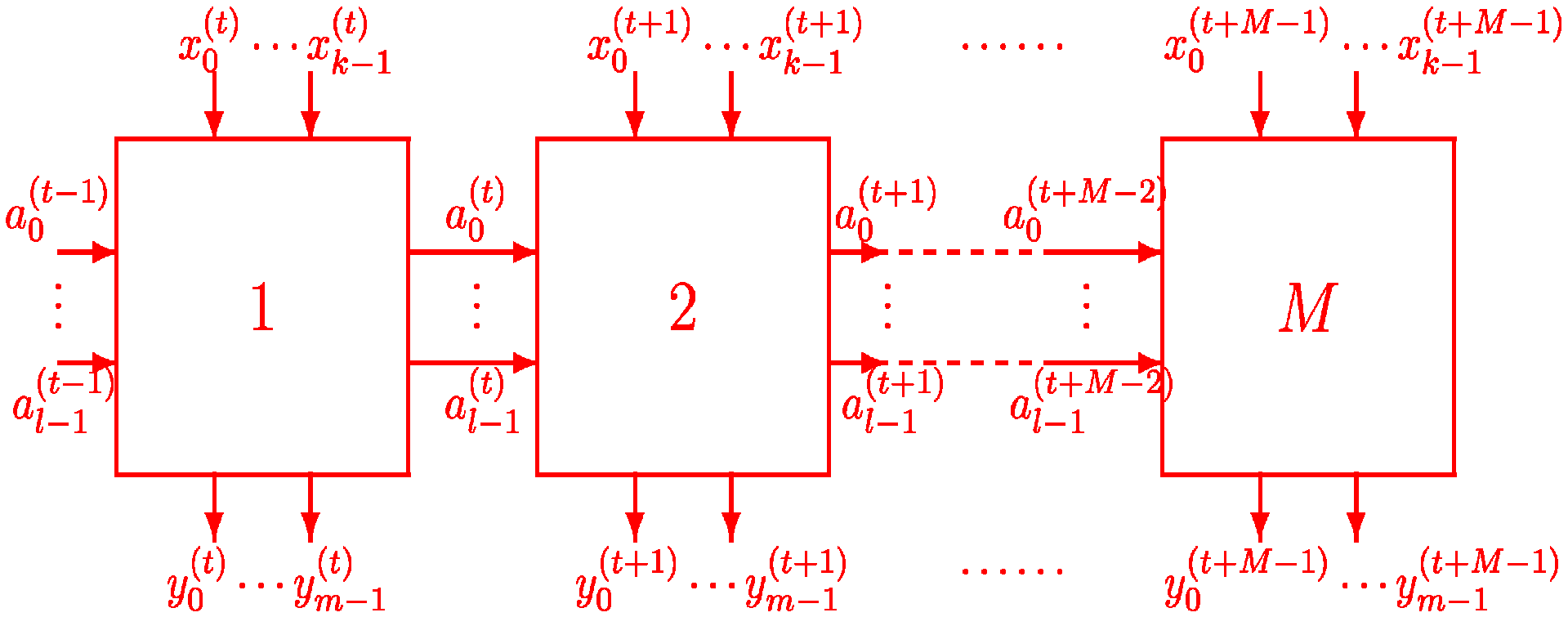
Goal:

Worse-case algebraic attacks: work for any LFSR-based stream cipher and **any** component of a given size.

We mathematically prove the existence of some algebraic equations that allow to break the cipher.

In some cases we ignore how these equations are found in practice (there may be special ad-hoc methods depending on the description of the cipher, and also in many other cases it is doable algebraically).

An LFSR-based Cipher with Memory



M successive applications of a combiner with k inputs, m outputs and l bits of memory

Our Main Result

This paper:
extremely strong general result in stream cipher cryptanalysis :

If the component is fixed, all these ciphers are broken in polynomial time (in the size of the LFSR). [for 1 output already proven by Krause-Armknecht]

In practice: usually impractical attacks.
Theoretical interest.

Our Main Theorem

Theorem 5.1. [Our Key Theorem]
Let F be an arbitrary fixed circuit/component with k binary inputs x_i, l bits of memory a_i, and m outputs y_i. Let d and M be two integers such that:

$$2^{Mm} \cdot \sum_{i=0}^{d} \binom{Mk}{i} > 2^{Mk+l} \qquad \textbf{(KE)}$$

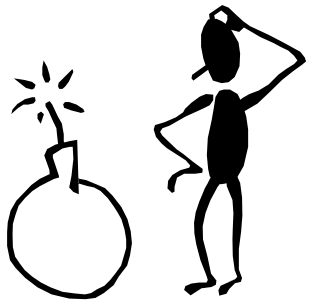
Then, considering M consecutive steps/states $(t, \ldots t+M-1)$, there is a multivariate equation (and relation) R of degree d in the $x_j^{(i)}$, relating the input and the output bits for these states

$$R\left(x_0^{(t)}, \ldots, x_{k-1}^{(t)}, \ldots, x_0^{(t+M-1)}, \ldots, x_{k-1}^{(t+M-1)};\right.$$
$$\left. y_0^{(t)}, \ldots, y_{m-1}^{(t)}, \ldots y_0^{(t+M-1)}, \ldots, y_{m-1}^{(t+M-1)}\right) = 0.$$

Remark:

There are many interesting ways of using this Theorem.

We discuss <u>ONLY few of them.</u>



Corollary 1:

Theorem 7.4.
[Generalised Krause-Armknecht Theorem]

Let F be an arbitrary fixed circuit/component with k binary inputs, l bits of memory, and m outputs.

Then, considering $M = \lceil (l+1)/m \rceil$ consecutive steps/states there is an algebraic attack with equations of degree , $\lceil kM/2 \rceil = \lceil k \lceil (l+1)/m \rceil /2 \rceil$

Corollary 2, Let m=1:

[Krause-Armknecht Theorem, Crypto 2003]

Let F be an arbitrary fixed circuit/component with k binary inputs, l bits of memory, and $m = 1$ outputs.

Then, considering $M = l+1$ consecutive steps/states there is an algebraic attack with equations of degree , $\lceil kM/2 \rceil = \lceil k(l+1)/2 \rceil$

Corollary 3, Let m=1, l=0:

[Courtois-Meier Theorem, Eurocrypt 2003]

Let F be a Boolean function (i.e. k binary inputs, 0 bits of memory, and $m = 1$ outputs.)

Then, considering $M = 1$ consecutive steps/states there is an algebraic attack with equations of degree , $\lceil kM/2 \rceil = k/2$

Proof:

Very simple, once you've done it.

Construct a system of monomials and show that they live in a linear space of other monomials, the dimension of which is smaller than the cardinal of it.

Therefore a dependency must exist.

See the paper for details.

We also prove a stronger Theorem in one case [cf. Appendix of the paper].

Applications – LILI-128 with more outputs

Table 1. Generic attacks on modified LILI-128 with m outputs

m	1	2	3	5	7
M	1	1	1	1	1
d	5	4	3	2	1

keystream	2^{25}	2^{64}	2^{21}	2^{60}	2^{17}	2^{56}	2^{12}	2^{51}	$\mathbf{2^{6}}$	$\mathbf{2^{45}}$
time(Step 1.)	2^{25}	2^{25}	2^{25}	2^{25}	2^{25}	2^{25}	2^{25}	2^{25}	2^{25}	2^{25}
time(Step 2.)	2^{107}	2^{68}	2^{95}	2^{56}	2^{83}	2^{44}	2^{69}	2^{30}	$\mathbf{2^{54}}$	$\mathbf{2^{15}}$

E0-type ciphers with more outputs

Table 2. Generic attacks on modified E0 with m outputs

m	1	2	3	4	5	6
M	5	3	2	3	1	1
d	10	5	3	2	2	1
keystream	2^{48}	2^{28}	2^{18}	2^{13}	2^{13}	$\mathbf{2^{7}}$
time(Step 1.)	2^{64}	2^{42}	2^{30}	2^{30}	2^{19}	$\mathbf{2^{19}}$
time(Step 2.)	2^{131}	2^{76}	2^{49}	2^{33}	2^{33}	$\mathbf{2^{16}}$

Snow-type ciphers with more outputs

Table 3. Generic attacks on modified Snow ciphers with m outputs

Snow 1.0. $n = 512, l = 64, k = 64$						
m	32	64	65	80	100	120
M	3	2	1	1	1	1
d	54	16	32	16	7	2
keystream	2^{245}	2^{99}	2^{169}	2^{99}	2^{51}	2^{17}
time(Step 1.)	2^{715}	2^{536}	2^{356}	2^{356}	2^{356}	2^{356}
time(Step 2.)	2^{684}	2^{276}	2^{471}	2^{276}	2^{139}	$\mathbf{2^{45}}$

And for modified Snow 2:

Table 3. Generic attacks on modified Snow ciphers with m outputs

Snow 2.0. $n = 512, l = 64, k = 96$					
m	32	64	65	120	150
M	3	2	1	1	1
d	92	35	48	9	2
keystream	2^{344}	2^{352}	2^{226}	2^{62}	2^{17}
time(Step 1.)	2^{985}	2^{715}	2^{446}	2^{446}	2^{446}
time(Step 2.)	2^{962}	2^{503}	2^{631}	2^{172}	$\mathbf{2^{45}}$

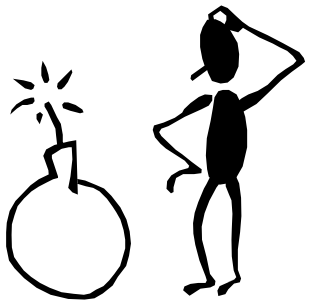
Summary of Our Results

- Many very general constructions of
- ! LFSR-based stream ciphers are broken in polynomial time and this for any component of a fixed size. Even if it has memory bits.
- In practice usually impractical… but: ciphers that output several bits at a time (in order to be fast) can be again be much weaker and broken.

Remark:

These attack work for any cipher of this "shape", worst case generic attacks.

For a specific cipher, much better attacks may exist !



You cannot claim that Snow or sth. is secure against algebraic attacks.

Requires proof.

Part 6

Further Work

Open question 36:

Given any component of a fixed size (with memory), and a LFSR of variable size.

Courtois [ICISC 2004]: broken in P time.

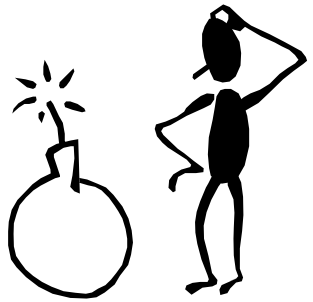
Conjecture [Courtois]: holds also when the component is secret [work in progress].

Part 7

General Conclusions

Ciphers with Linear Feedback:

Prone to algebraic attacks…



Can still use LFSR if we have either:

- Complex clocking/decimation
- Large state that is not updated linearly.

Remark

Little hope for Algebraic Attacks when one uses a lot of clock control / decimation…

For ciphers that are clocked in a “not so complex way”…

What happened to LFSR-based stream ciphers ?

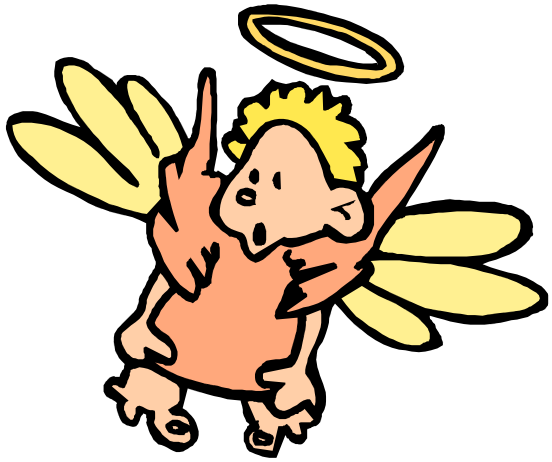
Some have been broken…

In general hard to determine if known algebraic attacks will not work !
(cf. Fast Alg. Attacks)

Need for provable security.
How ? OPEN PROBLEM.

Conclusions

"Highly non linear" Boolean functions are necessary,
! but not sufficient to build secure ciphers.



- Complex Boolean functions are not enough.
- Avoid also "algebraic relations".
- Boolean functions with high Algebraic Immunity guarantee very little (other existing "outputs" may destroy AI !).