Key Feedback Mode: a Keystream Generator with Provable Security

Johan Håstad
Royal Inst. of Technology, Sweden
Institute for Advanced Study

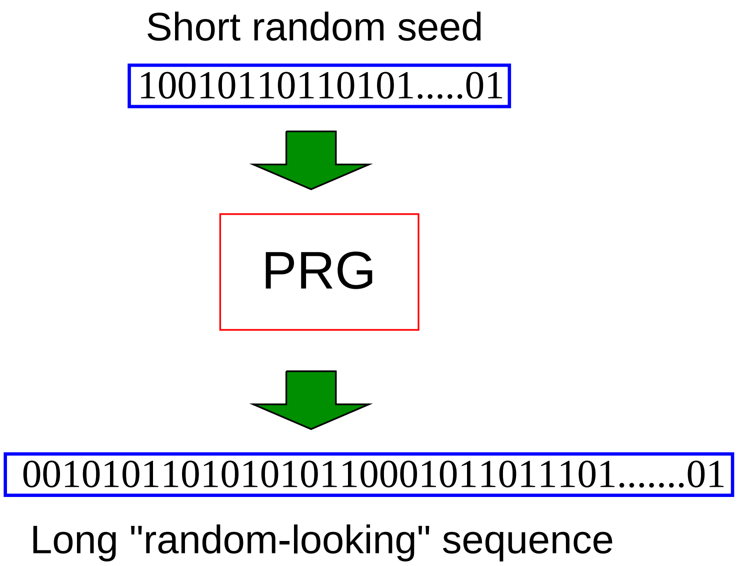
Mats Näslund
Ericsson Research, Sweden

The setup

Given A good block cryptosystem (AES).

Wanted A good pseudorandom generator.

A good pseudorandom generator



Short random seed

10010110110101.....01

PRG

001010110101010110001011011101.......01

Long "random-looking" sequence

- Generates many random looking bits from a short initial seed.

- Looks very similar to truly random bits. Passes many statistical tests.

Which statistical tests?

Classically A list of good standard tests.

Blum-Micali, Yao: All tests that can be implemented efficiently.

Passing a statistical test

P_R Probability the test rejects a truly random string.

P_G Probability the test rejects a string which is the output of the generator (on a random seed).

$|P_R - P_G| \geq \epsilon \quad \Leftrightarrow \quad \epsilon$-distinguishes.

How good is AES?

1. Hard to crack given only the cryptotext.

2. Hard to find the key given both the plaintext and the cryptotext.

3. Looks like a random permutation when the key is unknown.

The easy solution

Assume $AES_K(\cdot)$ behaves like a random permutation.

Counter-mode, i.e. outputting

$$AES_K(ctr + i), i = 0, 1, 2 \ldots$$

gives a good pseudorandom generator which is very efficient and (almost by definition) passes all statistical tests.

Our proposal

Assume that it is hard to find the key given the plaintext and the ciphertext (a diamond in the raw).

Cut and polish it to get a good pseudorandom generator.

Traditional academic set-up

We have a function f which is one-way, i.e. can be computed efficiently but cannot be inverted efficiently.

We construct a pseudorandom generator G_f with provable properties.

A sequence of constructions

Blum-Micali (82)	f discrete exponentiation.
Yao (82)	f any one-way permutation.
Goldreich-Levin (88)	f any one-way permutation, more efficient construction.
HILL (90)	f any one-way function.

The BMGL generator

Let f be a one-way permutation mapping $\{0,1\}^n$ to $\{0,1\}^n$.

Let x^0 be random initial seed and r a random string in $\{0,1\}^n$.

$x^i = f(x^{i-1})$

$b^i = \langle r, x^i \rangle = \sum_{j=1}^{n} r_j x_j^i \bmod 2$

Output of G_f is $b_1, b_2, b_3 \ldots$

Keep iterating f and outputting the xor of the bits defined by the vector r.

Properties of the BMGL generator

- If f is a permutation that cannot be inverted in polynomial time then the output of G_f cannot be distinguished from truly random bits in polynomial time.

- If f is a one-way function which remain one-way even when iterated, then the same conclusion is true. [Levin]

In our case

Our one-way function

$$f_{AES}(x) = AES(x, P)$$

for any fixed plaintext message P. Note, input is KEY, output is normal output (i.e. the ciphertext).

Blocksize=keysize

Properties of our one-way function

Assumption: cannot be inverted much faster than exhaustive search.

It is probably not a permutation, although we do not know this for sure.

A tempting alternative

The encryption function

$$f_{AES}(x) = AES(K, x).$$

It is a permutation (good).

It is as easy to invert as to compute (bad).

We get no provable properties, no source of contradiction.

Problems to discuss

Moving from “polynomial time” to concrete security. Assume that close to 2^{256} encryptions are required to invert f. Need to optimize proofs and chase constants.

Limit time of statistical tests to something concrete (most natural tests are faster than the generator).

Speed of generator is bad. Only outputs one bit for a full block-encryption.

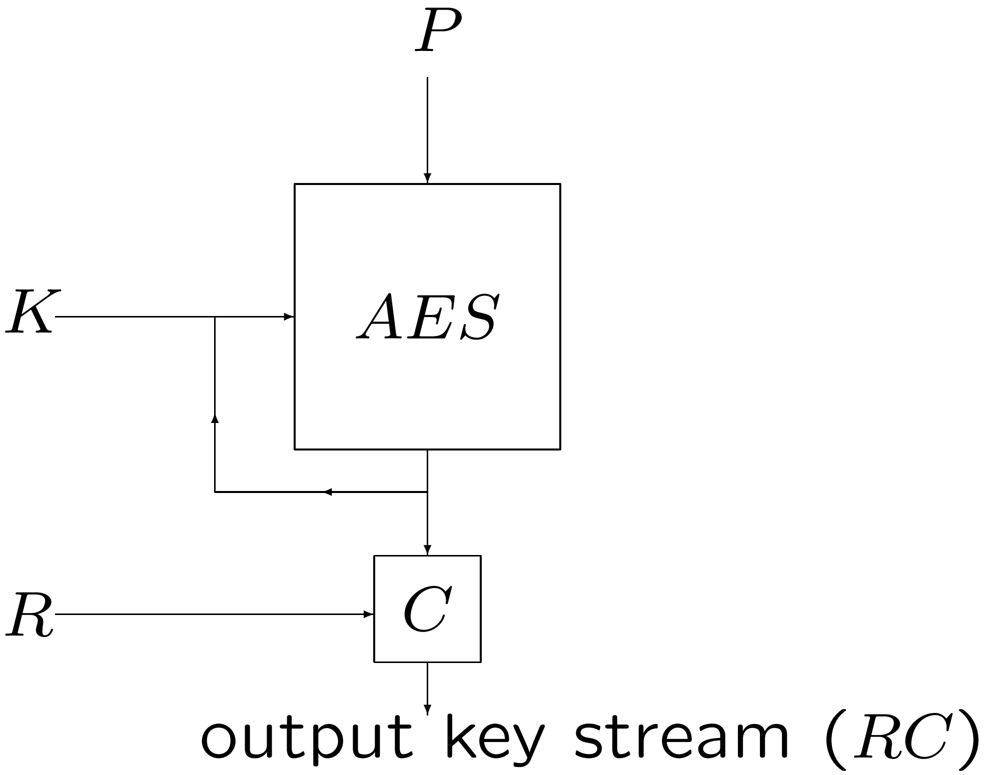
Improving speed

We can output more bits for each iteration.

- Output Rx^j for an $m \times n$ matrix R. Either random or Toeplitz (constant on diagonals).

- Compute ax^j in $GF[2^n]$ and output any m bits. A random $a \in GF[2^n]$.

The final BMGL-generator



Key Feedback Mode.

Type of assumption needed

It is hard to invert f_{AES}, not only in the ordinary sense but it remains hard when iterated.

Type of conclusion wanted

No statistical test using time less than 2^{30} evaluations of AES can guess whether a string is truly random or output of BMGL with probability greater than $\frac{1}{2} + 2^{-30}$ of being correct.

Details of assumptions

Let $f^{(i)}$ be f iterated i times, i.e.

$$f^{(i)}(x) = \overbrace{f(f(..f}^{i}(x)..)).$$

i-inverting: Given $y = f^{(i)}(x)$ for random x, find z such that $f(z) = y$.

Theorem: A random f cannot be i-inverted faster than $\approx 2^n/i$ on n bit strings.

ϵ strong function

A one-way function f is ϵ strong if it cannot be i-inverted faster than $\epsilon 2^n/i$ evaluations of f.

A random function is about 1-strong.

We expect f_{AES} to be ϵ-strong for a reasonably large ϵ.

A concrete theorem

Assume that f_{AES} on 256-bit blocks is 2^{-25}-strong.

Output 40 bits for each iteration for a total output of 2^{30} bits.

Statistical test has running time is at most 2^{25} applications of AES.

Cannot guess whether a string is random or output from the BMGL-generator and be correct with probability $\geq .500000001$.

Another concrete theorem

Assume that f_{AES} on 256-bit blocks is 2^{-32}-strong.

Output 4 bits for each iteration for a total output of 2^{30} bits.

Statistical test has running time is at most 2^{28} applications of AES.

Cannot guess whether a string is random or output from the BMGL-generator and be correct with probability $\geq$ .500000000001.

One small problem

Seed size. Theorems as stated requires a random matrix of size $m \times n$, (40×256).

Accepting a speed decrease by a factor 2, can reduce seed size from $256m$ to 256 bits maintaining security.

Summary

- Key feed back mode. Keep the same fixed plaintext. Feed output as key for next round.

- Output a simple function of the ciphertext at each iteration.

- Behaves provably as random bits unless the function becomes easy on its iterates.