# $p$-Adic Dynamical Systems and Cryptography

## *Non-Archimedean View on $T$-functions*

Vladimir Anashin

Russian State University for the Humanities
Faculty of Information Security

# $T$-functions: Where they come from?

In 2002 Klimov and Shamir introduced to crypto community a class of mappings they called *$T$-functions*:

$$(\alpha_0^{\downarrow}, \alpha_1^{\downarrow}, \alpha_2^{\downarrow}, \ldots) \mapsto (\Phi_0(\alpha_0^{\downarrow}), \Phi_1(\alpha_0^{\downarrow}, \alpha_1^{\downarrow}), \Phi_2(\alpha_0^{\downarrow}, \alpha_1^{\downarrow}, \alpha_2^{\downarrow}), \ldots).$$

Here $\alpha_i^{\downarrow} \in \mathbb{B}^m$ is a Boolean columnar $m$-dimensional vector; $\mathbb{B} = \{0, 1\}$; $\Phi_i \colon (\mathbb{B}^m)^{(i+1)} \to \mathbb{B}^n$ maps $(i+1)$ Boolean columnar $m$-dimensional vectors $\alpha_0^{\downarrow}, \ldots, \alpha_i^{\downarrow}$ to $n$-dimensional Boolean vector $\Phi_i(\alpha_0^{\downarrow}, \ldots, \alpha_i^{\downarrow})$.

These mappings are of interest for software-oriented ciphers, since both arithmetic and bitwise logical operations, which are basic instructions for most processors, are obviously $T$-functions.

# $T$-functions: Where they come from?

In 2002 Klimov and Shamir introduced to crypto community a class of mappings they called *$T$-functions*:

*In mathematics these mappings are known more than 30 years. The mathematical theory of these mappings is well developed.*
In different areas of mathematics these mappings were studied under different names, for instance:

- *Triangle mappings*, in the theory of Boolean functions;

# $T$-functions: Where they come from?

In 2002 Klimov and Shamir introduced to crypto community a class of mappings they called *$T$-functions*:

*In mathematics these mappings are known more than 30 years. The mathematical theory of these mappings is well developed.*

In different areas of mathematics these mappings were studied under different names, for instance:

- *Triangle mappings*, in the theory of Boolean functions;
- *Determined functions*, in automata theory;

# $T$-functions: Where they come from?

In 2002 Klimov and Shamir introduced to crypto community a class of mappings they called *$T$-functions*:

*In mathematics these mappings are known more than 30 years. The mathematical theory of these mappings is well developed.*
In different areas of mathematics these mappings were studied under different names, for instance:

- *Triangle mappings*, in the theory of Boolean functions;
- *Determined functions*, in automata theory;
- *Compatible functions on the residue ring $\mathbb{Z}/2^n$*, in algebra, etc.

# $T$-functions: Where they come from?

In early 90-th the non-Archimedean theory of $T$-functions started, which treated $T$-functions as continuous mappings of the space $\mathbb{Z}_2$ of 2-adic integers.

# $T$-functions: Where they come from?

In early 90-th the non-Archimedean theory of $T$-functions started, which treated $T$-functions as continuous mappings of the space $\mathbb{Z}_2$ of 2-adic integers.

In our talk we introduce some methods and results from this theory. These methods are of use for

- the design of fast and flexible stream ciphers based on $T$-functions, and for
- the study of important properties of these ciphers.

# Stream encryption is easy!

- Take a plain text $\alpha_0 \quad \alpha_1 \quad \alpha_2 \quad \ldots$

# Stream encryption is easy!

- Take a plain text $\alpha_0 \quad \alpha_1 \quad \alpha_2 \quad \ldots$
- add it modulo 2 $\oplus$

# Stream encryption is easy!

- Take a plain text $\alpha_0 \quad \alpha_1 \quad \alpha_2 \quad \ldots$
- add it modulo 2 $\oplus$
- to a key stream $\gamma_0 \quad \gamma_1 \quad \gamma_2 \quad \ldots$

# Stream encryption is easy!

- Take a plain text $\alpha_0 \quad \alpha_1 \quad \alpha_2 \quad \ldots$
- add it modulo 2 $\oplus$
- to a key stream $\gamma_0 \quad \gamma_1 \quad \gamma_2 \quad \ldots$
- and get

# Stream encryption is easy!

- Take a plain text $\alpha_0 \quad \alpha_1 \quad \alpha_2 \quad \ldots$
- add it modulo 2 $\oplus$
- to a key stream $\gamma_0 \quad \gamma_1 \quad \gamma_2 \quad \ldots$
- and get
- encrypted text $\zeta_0 \quad \zeta_1 \quad \zeta_2 \quad \ldots$

# Stream encryption is easy!

To decrypt, take

- the encrypted text $\zeta_0 \quad \zeta_1 \quad \zeta_2 \quad \ldots$

# Stream encryption is easy!

To decrypt, take

- the encrypted text $\zeta_0 \quad \zeta_1 \quad \zeta_2 \quad \ldots$
- add it modulo 2 $\oplus$

# Stream encryption is easy!

To decrypt, take

- the encrypted text $\zeta_0 \quad \zeta_1 \quad \zeta_2 \quad \ldots$
- add it modulo 2 $\oplus$
- to the key stream $\gamma_0 \quad \gamma_1 \quad \gamma_2 \quad \ldots$

# Stream encryption is easy!

To decrypt, take

- the encrypted text
- add it modulo 2
- to the key stream
- and get

$$\begin{array}{lcccc} & \zeta_0 & \zeta_1 & \zeta_2 & \dots \\ \oplus & & & & \\ & \gamma_0 & \gamma_1 & \gamma_2 & \dots \\ \hline \end{array}$$

# Stream encryption is easy!

To decrypt, take

- the encrypted text
- add it modulo 2
- to the key stream
- and get
- the plain text

$$\begin{array}{lllll} & \zeta_0 & \zeta_1 & \zeta_2 & \dots \\ \oplus & & & & \\ & \gamma_0 & \gamma_1 & \gamma_2 & \dots \\ \hline & \alpha_0 & \alpha_1 & \alpha_2 & \dots \end{array}$$

*Shannon's Theorem yields that the encryption is secure whenever one chooses key stream at random.*

*Shannon's Theorem yields that the encryption is secure whenever one chooses key stream at random.*
And Kolmogorov's complexity theory says that it is impossible to produce a random sequence by a deterministic algorithm.

*Shannon's Theorem yields that the encryption is secure whenever one chooses key stream at random.*
And Kolmogorov's complexity theory says that it is impossible to produce a random sequence by a deterministic algorithm.

Could we use stream encryption on computers in a way other than to store huge amounts of key stream bits on hard drives?

# We could.

# We could.

Given a family $\mathcal{T}$ of statistical tests, a *pseudorandom* sequence (with respect to $\mathcal{T}$) is the one that passes all the tests of $\mathcal{T}$.

# We could.

Given a family $\mathcal{T}$ of statistical tests, a *pseudorandom* sequence (with respect to $\mathcal{T}$) is the one that passes all the tests of $\mathcal{T}$.

Assuming an *adversary* can use only the tests of $\mathcal{T}$, he can not distinguish a pseudorandom sequence from a truly random one.

# We could.

Given a family $\mathcal{T}$ of statistical tests, a *pseudorandom* sequence (with respect to $\mathcal{T}$) is the one that passes all the tests of $\mathcal{T}$.

Assuming an *adversary* can use only the tests of $\mathcal{T}$, he can not distinguish a pseudorandom sequence from a truly random one. That is, an adversary can not decrypt the message whenever a key stream is pseudorandom.

## We could.

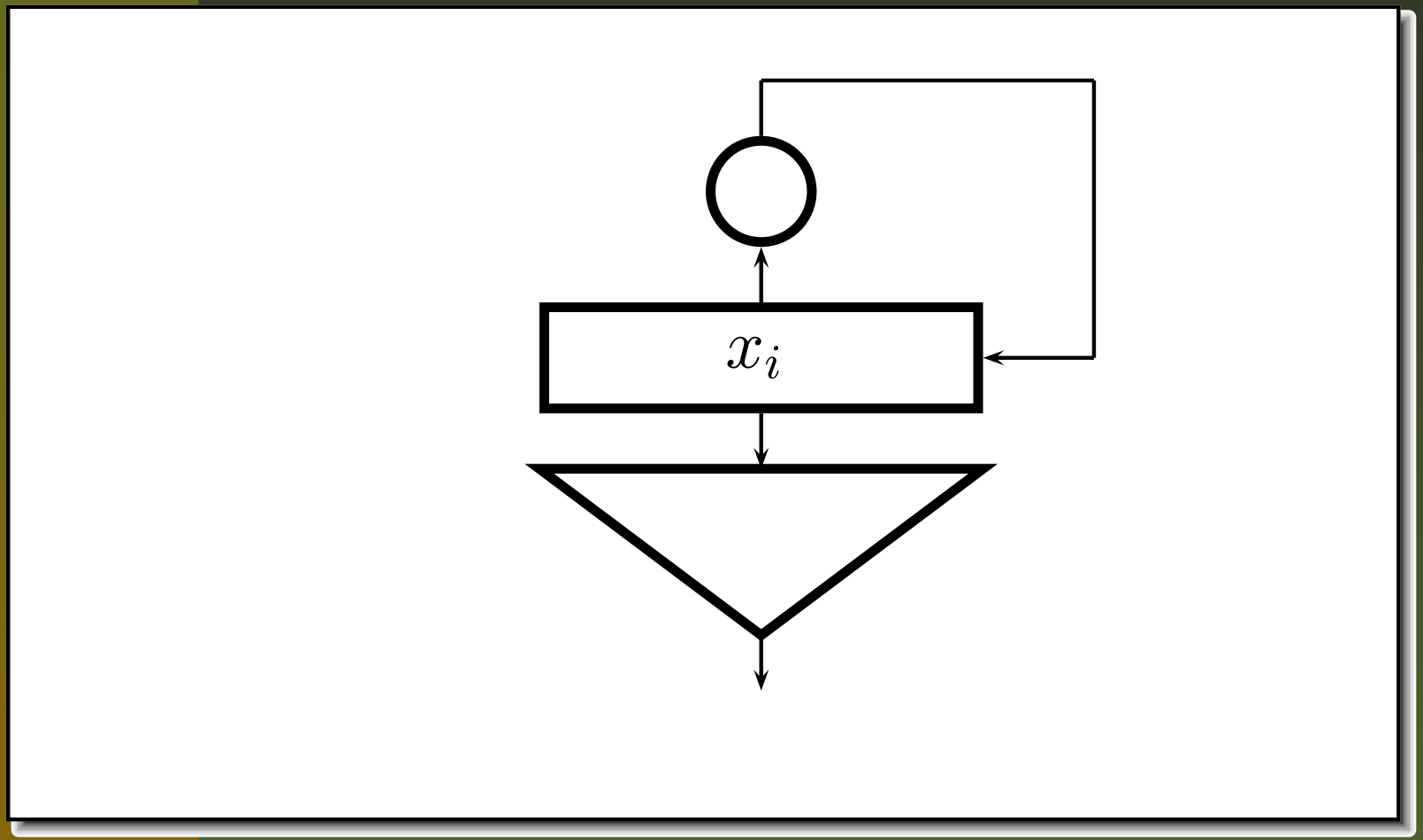
Given a family $\mathcal{T}$ of statistical tests, a *pseudorandom* sequence (with respect to $\mathcal{T}$) is the one that passes all the tests of $\mathcal{T}$.

Assuming an *adversary* can use only the tests of $\mathcal{T}$, he can not distinguish a pseudorandom sequence from a truly random one. That is, an adversary can not decrypt the message whenever a key stream is pseudorandom.

## It is possible to produce a pseudorandom sequence by an algorithm, under some reasonable choices of $\mathcal{T}$.

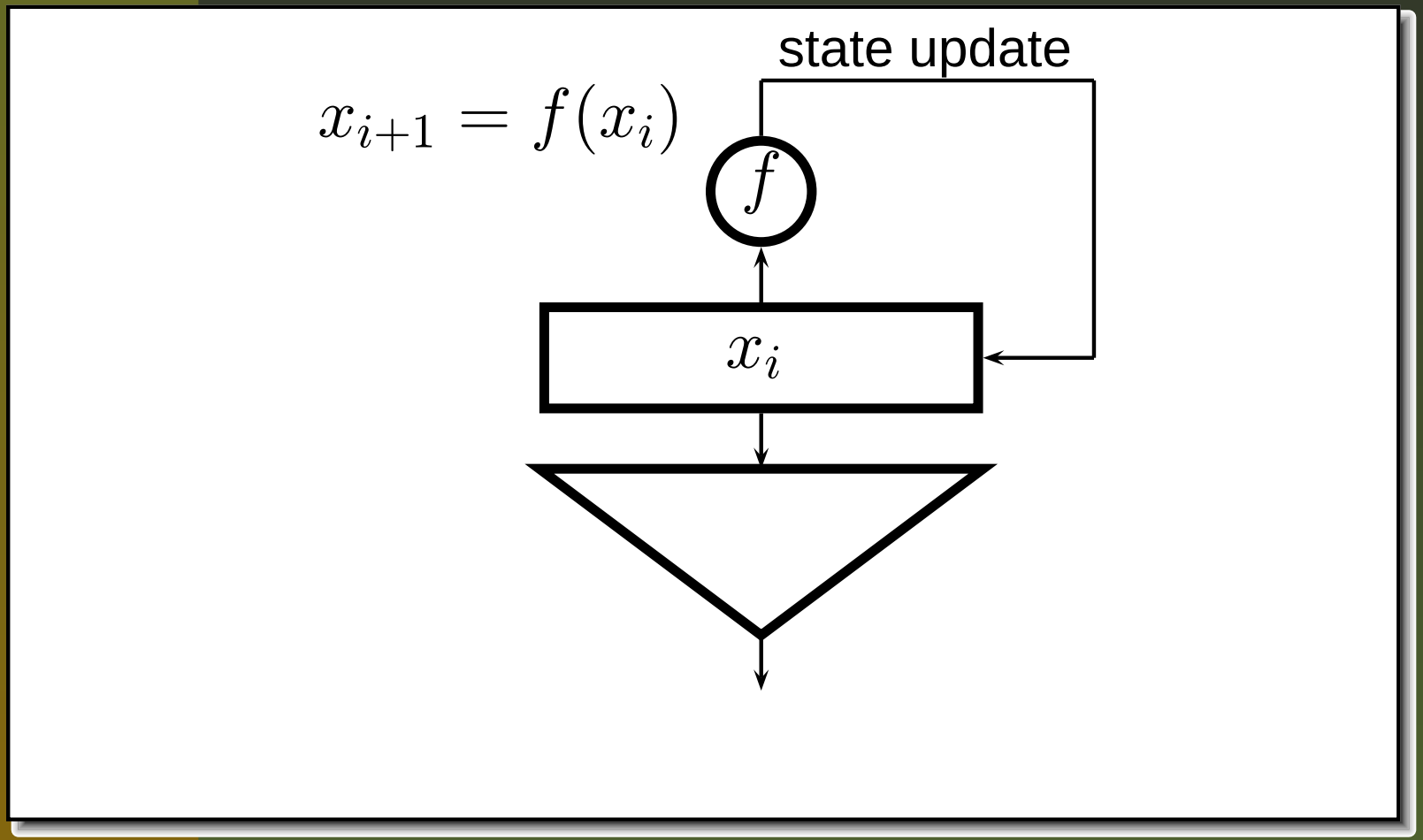
# Pseudorandom number generator

PRNG produces a key stream.



# Pseudorandom number generator

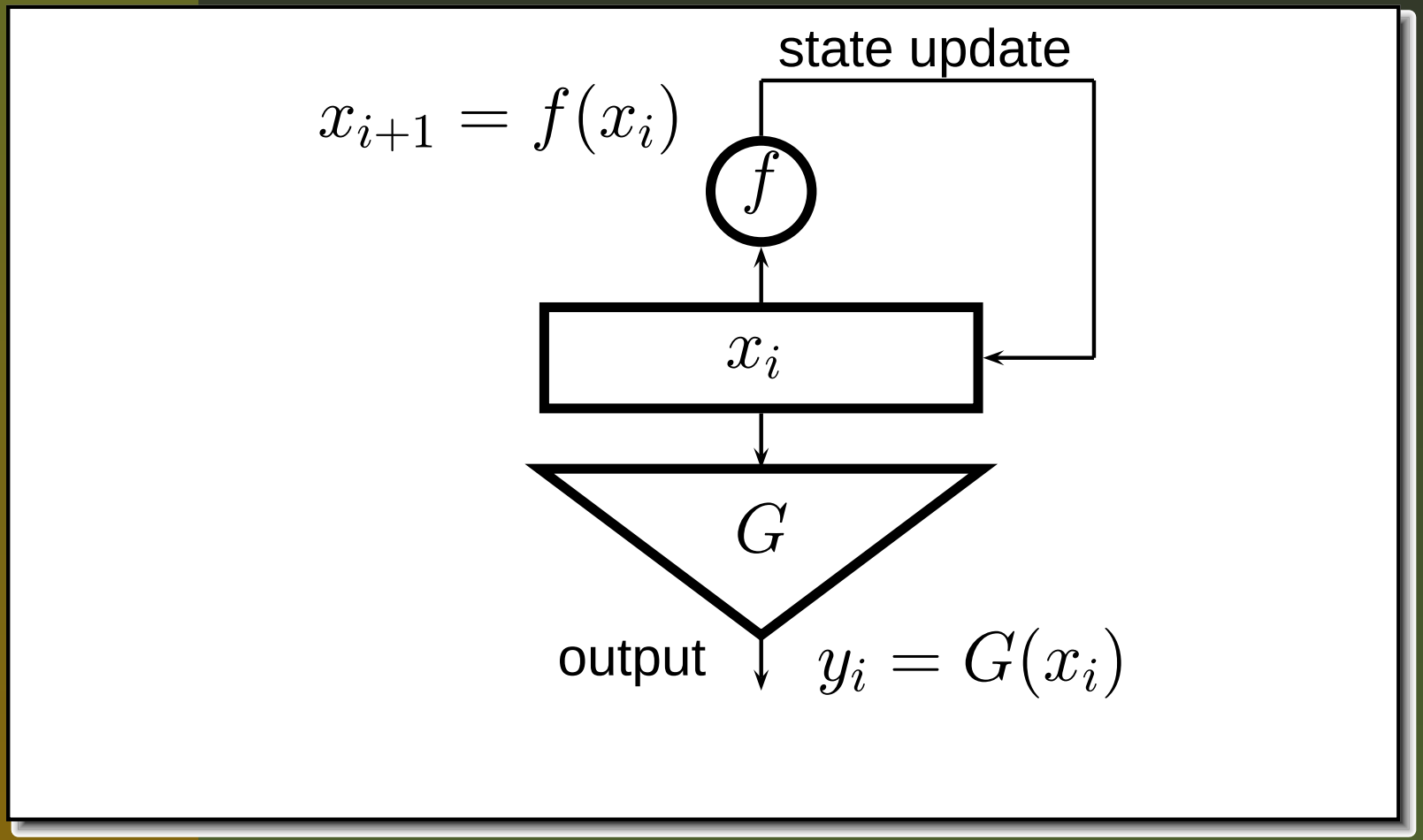
PRNG produces a key stream.




$f : A \to A$ is the *state update* function,

# Pseudorandom number generator

PRNG produces a key stream.




$f : A \to A$ is the *state update* function, $G : A \to B$ is the *output* function.

The sequence of internal states $\{x_i \in A\}$ of the PRNG is the sequence

$$x_0,\ x_1 = f(x_0), \ldots, x_{i+1} = f(x_i) = f^{i+1}(x_0), \ldots$$

The output sequence $\{y_i \in B\}$ satisfies the law $y_i = G(x_i)$, $(i = 0, 1, 2, \ldots)$.

In classical stream ciphers a *key* is the initial state $x_0$.

A key is the only information that is not known to an adversary.

Most often the set of internal states (the internal alphabet) $A$ is the set $\mathbb{B}^n$ of all n-bit words; the output alphabet $B$ is the set $\mathbb{B}^k$ of all $k$-bit words.

It is convenient to associate the set $\mathbb{B}^n$ to the residue ring $\mathbb{Z}/2^n$ up to the natural one-to-one correspondence:

To each $z \in \mathbb{Z}/2^n = \{0, 1, 2, \ldots, 2^n - 1\}$ there corresponds one and the only $n$-bit word of $\mathbb{B}^n$, which is a base-2 expansion of $z$.

$$\mathbb{Z}/2^n \ni z = \zeta_0 + \zeta_1 \cdot 2 + \zeta_2 \cdot 2^2 + \cdots \longleftrightarrow \zeta_0\zeta_1\zeta_2 \ldots \in \mathbb{B}^n$$

# Why dynamical systems?

An autonomous dynamical system is a suite $\langle \mathfrak{X}, \mu, \mathbf{f} \rangle$, where $\mathfrak{X}$ is a phase space (usually a metric space), $\mu$ is a measure on $\mathfrak{X}$ (e.g., probabilistic one); $\mathbf{f} : \mathfrak{X} \to \mathfrak{X}$ is a measurable mapping (usually, continuous).

A trajectory of the point $\mathbf{x}_0$ is a sequence

$$\mathbf{x}_0,\ \mathbf{x}_1 = \mathbf{f}(\mathbf{x}_0), \ldots, \mathbf{x}_{i+1} = \mathbf{f}(\mathbf{x}_i) = \mathbf{f}^{i+1}(\mathbf{x}_0), \ldots .$$

Dynamical systems theory prompts a very natural approach: Let $\langle \mathfrak{X}, \mu, \mathbf{f} \rangle$ be a dynamical system with discrete time. Take a point $\mathbf{x}_0 \in \mathfrak{X}$ as a key, and use the trajectory as a source of pseudorandomness.

To make this approach to stream cipher design meaningful, the following questions must be answered:

- How one could evaluate the trajectory on a digital computer?

To make this approach to stream cipher design meaningful, the following questions must be answered:

- How one could evaluate the trajectory on a digital computer?
- What will be the performance?

To make this approach to stream cipher design meaningful, the following questions must be answered:

- How one could evaluate the trajectory on a digital computer?
- What will be the performance?
- How pseudorandom is the so produced sequence?

To make this approach to stream cipher design meaningful, the following questions must be answered:

- How one could evaluate the trajectory on a digital computer?
- What will be the performance?
- How pseudorandom is the so produced sequence?
- Is the corresponding generator secure?

# Any use of chaos?

Since early 90$^{th}$ intensive studies were undertaken in the *chaos-based cryptography*.

The leading idea of the latter is quite natural:
Take a chaotic map $\mathbf{f}$ and make it discrete!

The trajectory will hopefully look like random since the mapping is chaotic (that is, sensitive to small perturbations of the initial state).

# Bad news

Results of such a straightforward approach turned out to be rather disappointing:

# Bad news

Results of such a straightforward approach turned out to be rather disappointing:

*Example.* A discrete version of the doubling map (Bernoulli shift) $\mathbf{f}(\mathbf{x}) = (2 \cdot \mathbf{x}) \bmod 1$ is $x_{i+1} \equiv 2 \cdot x_i \pmod{2^n}$ becomes $0$ after at most $n$ iterations!!!

# Bad news

Results of such a straightforward approach turned out to be rather disappointing:

*Example.* A discrete version of the doubling map (Bernoulli shift) becomes $0$ after at most $n$ iterations!!!

*One more example.* A discrete version of the tent map $\mathbf{f}(\mathbf{x}) = 1 - 2 \cdot |\mathbf{x} - \frac{1}{2}|$ on $[0, 1]$ always falls in very short cycles, of length $n$ at most!!!

# Bad news

Results of such a straightforward approach turned out to be rather disappointing:

*Example.* A discrete version of the doubling map (Bernoulli shift) becomes $0$ after at most $n$ iterations!!!

*One more example.* A discrete version of the tent map always falls in very short cycles, of length $n$ at most!!!

*Yet another example.* A discrete version of the logistic map $\mathbf{f}(\mathbf{x}) = 4 \cdot \mathbf{x} \cdot (1 - \mathbf{x}) \bmod 1$ becomes $0$ after at most $\frac{n}{2}$ iterations!!!

## **L. Kocarev.** *Chaos-Based Cryptography: A Brief Overview* **(2001):**

Despite a huge number of papers published in the field of chaos-based cryptography, the impact that this research has made on conventional cryptography is *rather marginal*. This is due to two reasons:

- First, almost all chaos-based cryptographic algorithms use dynamical systems defined on the set of *real numbers*, and therefore *are difficult for practical realization* and circuit implementation.

## **L. Kocarev.** *Chaos-Based Cryptography: A Brief Overview* **(2001):**

Despite a huge number of papers published in the field of chaos-based cryptography, the impact that this research has made on conventional cryptography is *rather marginal*.

- First, almost all chaos-based cryptographic algorithms *are difficult for practical realization* and circuit implementation.
- Second, security and performance of almost all proposed chaos-based methods are not analyzed in terms of the techniques developed in cryptography. Moreover, most of the *proposed methods generate cryptographically weak and slow algorithms.*

## **Shujun Li.** *When Chaos Meets Computers* **(2004):**

Digital computers are absolutely incapable of showing true long-time dynamics of some chaotic systems, including the tent map, the Bernoulli shift map and their analogues, even in a high-precision floating-point arithmetic. Although the results cannot directly generalized to most chaotic systems, the risk of using digital computers to numerically study continuous dynamical systems is shown clearly. As a result, we reach the old saying that “it is impossible to do everything with computers only”.

Despite these pessimistic conclusions of the two of key researchers of chaos-based cryptography, there are very promising developments in stream cipher design related to dynamical systems theory.

Despite these pessimistic conclusions of the two of key researchers of chaos-based cryptography, there are very promising developments in stream cipher design related to dynamical systems theory.

Surprisingly, these developments are related neither to real nor to complex, but to the non-Archimedean dynamical systems theory!

# What is a good PRNG?

A cryptographic PRNG must meet the following conditions:

- For (almost) all keys the output sequences must be pseudorandom (i.e., undistinguishable from a truly random sequence up to the tests of $\mathcal{T}$).

# What is a good PRNG?

A cryptographic PRNG must meet the following conditions:

- For (almost) all keys the output sequences must be pseudorandom (i.e., undistinguishable from a truly random sequence up to the tests of $\mathcal{T}$).
- Given a segment $y_j, y_{j+1}, \ldots, y_{j+s-1}$ of the output sequence, finding the corresponding key must be infeasible (in some properly defined sense).

# What is a good PRNG?

A cryptographic PRNG must meet the following conditions:

- For (almost) all keys the output sequences must be pseudorandom (i.e., undistinguishable from a truly random sequence up to the tests of $\mathcal{T}$).
- Given a segment $y_j, y_{j+1}, \ldots, y_{j+s-1}$ of the output sequence, finding the corresponding key must be infeasible (in some properly defined sense).
- The PRNG must be suitable for software (or hardware) implementation; the performance must be sufficiently fast.

In other words:

- The *state update function $f$ must provide pseudorandomness*; in particular, it must guarantee uniform distribution and long period of the state update sequence $\{x_i\}$.

In other words:

- The *state update function $f$ must provide pseudorandomness*
- The *output function $G$ must not spoil the pseudorandomness* (in particular, the output sequence $\{y_i\}$ must be uniformly distributed and must have long period); and moreover, *$G$ must make the PRNG secure* (in particular, given $y_i$, it must be difficult to find $x_i$ from the equation $y_i = G(x_i)$).

In other words:

- The *state update function $f$ must provide pseudorandomness*
- The *output function $G$ must not spoil the pseudorandomness* ; and moreover, *$G$ must make the PRNG secure*
- To make the PRNG suitable for software/hardware implementations, both *$f$ and $G$ must be compositions of basic microprocessor instructions.*

# Designing PRNG

To satisfy condition 1 (of 3) a good secure PRNG must meet, one could take the state update function $f : \mathbb{Z}/2^n \to \mathbb{Z}/2^n$ with a *single cycle property*; that is, $f$ permutes elements of $\mathbb{Z}/2^n$ cyclically.

The state update sequence

$$x_0,\ x_1 = f(x_0), \ldots, x_{i+1} = f(x_i) = f^{i+1}(x_0), \ldots$$

of $n$-bit words will have then the longest possible period (of length $2^n$), and *strict uniform distribution*; that is, each $n$-bit word will occur at the period exactly once.

# Designing PRNG

To satisfy the first part of condition 2 one could take a *balanced* mapping $G\colon \mathbb{Z}/2^n \to \mathbb{Z}/2^k$.

That is, to each $k$-bit word the mapping $G$ maps the same number of $n$-bit words (hence; $k \leq n$). For $k = n$ balanced mappings are just *invertible* (that is, bijective, one-to-one) mappings.

For $k \ll n$ balanced functions could be of use to satisfy the second part of the condition 2, since the equation $y_i = G(x_i)$ has too many solutions then, $2^{n-k}$.

# Designing PRNG

To satisfy condition 3, one must know how to construct single cycle (respectively, balanced) mappings out of basic microprocessor instructions, which include:

- *integer arithmetic operations* (addition, multiplication,...)
- *bitwise logical operations* (OR, XOR, AND, NOT)
- *machine operations* (shifts, masking, sometimes cyclic shifts).

# Designing PRNG

To satisfy condition 3, one must know how to construct single cycle (respectively, balanced) mappings out of basic microprocessor instructions, which include:

- *integer arithmetic operations* (addition, multiplication,...)
- *bitwise logical operations* (OR, XOR, AND, NOT)
- *machine operations* (shifts, masking, sometimes cyclic shifts).

This could be done with the use of 2-adic analysis!

# More attentive look ...

Let $z = \delta_0(z) + \delta_1(z) \cdot 2 + \delta_2(z) \cdot 2^2 + \delta_3(z) \cdot 2^3 + \cdots$ be a base-2 expansion for $z \in \mathbb{N}_0$; then:

- $y \,\mathsf{XOR}\, z = y \oplus z$ is a bitwise addition modulo 2: $\delta_j(y \,\mathsf{XOR}\, z) \equiv \delta_j(y) + \delta_j(z) \pmod 2$;
- $y \,\mathsf{AND}\, z$ is a bitwise multiplication modulo 2: $\delta_j(y \,\mathsf{AND}\, z) \equiv \delta_j(y) \cdot \delta_j(z) \pmod 2$;
- $\left\lfloor \frac{z}{2} \right\rfloor$ is a shift towards less significant bits;
- $2 \cdot z$ is a shift towards more significant bits;
- $y \,\mathsf{AND}\, z$ is the masking of $z$ with the mask $y$;
- $z \pmod{2^k} = z \,\mathsf{AND}(2^k - 1)$ is a reduction of $z$ modulo $2^k$

# ... and tiny observations

All basic chip operations, with the only exception of cyclic shifts, are well defined on the space $\mathbb{Z}_2$ of all 2-adic integers.

# ... and tiny observations

All basic chip operations, with the only exception of cyclic shifts, are well defined on the space $\mathbb{Z}_2$ of all 2-adic integers.

The space $\mathbb{Z}_2$ could be thought of as a set of all countable infinite binary sequences.

# ... and tiny observations

All basic chip operations, with the only exception of cyclic shifts, are well defined on the space $\mathbb{Z}_2$ of all 2-adic integers.
The following example proves that $\ldots 11111 = -1$.

$$
\begin{array}{cccccc}
 & \ldots 1 & 1 & 1 & 1 \\
+ & & & & \\
 & \ldots 0 & 0 & 0 & 1 \\
\hline
 & \ldots 0 & 0 & 0 & 0
\end{array}
$$

# Do you know that $\ldots 1010101 = -\frac{1}{3}$?

$$
\begin{array}{cccccc}
\ldots 0 & 1 & 0 & 1 & 0 & 1 \\
\times & & & & & \\
\ldots 0 & 0 & 0 & 0 & 1 & 1 \\
\hline
\ldots 0 & 1 & 0 & 1 & 0 & 1 \\
+ & & & & & \\
\ldots 1 & 0 & 1 & 0 & 1 & \\
\hline
\ldots 1 & 1 & 1 & 1 & 1 & 1
\end{array}
$$

# **Do you know that** $\ldots 1010101 = -\frac{1}{3}$**?**

A calculator knows that either!

# A short 2-adic tour

Sequences with only finite number of 1's correspond to non-negative rational integers in their base-2 expansions:

$$\ldots 00011 = 3$$

# A short 2-adic tour

Sequences with only finite number of $0$'s correspond to negative rational integers:

$$\ldots 111100 = -4$$

# A short 2-adic tour

Eventually periodic sequences correspond to rational numbers represented by irreducible fractions with odd denominators:

$$\ldots 1010101 = -\frac{1}{3}$$

# A short 2-adic tour

Sequences that are not (eventually) periodic correspond to no rational number:

$$\ldots 011110111101101$$

# A short 2-adic tour

Distance: $d_2(u,v) = 2^{-k}$ iff

$$u \equiv v \pmod{2^k}; u \not\equiv v \pmod{2^{k+1}}$$

The longer are common initial segments of sequences the closer are the points!

The space $\mathbb{Z}_2$ is complete with respect to the 2-adic distance (metric) $d_2$, and compact.

# A short 2-adic tour

As usual, the norm is $\|u\|_2 = d_2(u, 0)$.

The higher power of 2 is a factor of a 2-adic integer the smaller the integer is!

# A short 2-adic tour

Once distance and norm are defined, notions of limits, convergent series, continuous functions, derivatives, etc., become meaningful:

$$d_2(-1,3) = \|(-1)-3\|_2 = \|-4\|_2 = \frac{1}{2^2} = \frac{1}{4};$$

$$\lim_{n\to\infty}{}^{d_2} 2^n = 0;$$

$$\ln(-3) = -\sum_{i=1}^{\infty} \frac{4^i}{i} \text{ is a 2-adic integer!}$$

# More observations

Basic chip operations (with the exception of cyclic shifts) are well defined continuous $\mathbb{Z}_2$-valued functions of 2-adic integer arguments.

# More observations

Basic chip operations (with the exception of cyclic shifts) are well defined continuous $\mathbb{Z}_2$-valued functions of 2-adic integer arguments.

Moreover, all mentioned functions (with the exception of those defined by shifts towards less significant bits) satisfy Lipschitz condition with coefficient 1 with respect to the 2-adic metric.

# More observations

Basic chip operations (with the exception of cyclic shifts) are well defined continuous $\mathbb{Z}_2$-valued functions of 2-adic integer arguments.

All compositions $F$ of basic chip instructions (with the exceptions of cyclic shifts, and shifts towards less significant bits) satisfy Lipschitz condition with coefficient 1:

$$\|F(a) - F(b)\|_2 \leq \|a - b\|_2$$

# Terminology notes

The condition

$$F(a) \equiv F(b) \pmod{2^k} \text{ whenever } a \equiv b \pmod{2^k}$$

is equivalent to the condition

$$\|F(a) - F(b)\|_2 \leq \|a - b\|_2$$

That is, $F$ satisfy Lipschitz condition with coefficient 1 iff $F$ is a *compatible* mapping of the ring $\mathbb{Z}_2$ into itself.

# Terminology notes

The condition

$$F(a) \equiv F(b) \pmod{2^k} \text{ whenever } a \equiv b \pmod{2^k}$$

is equivalent to the condition

$$\|F(a) - F(b)\|_2 \leq \|a - b\|_2$$

That is, $F$ satisfy Lipschitz condition with coefficient 1 iff $F$ is a *compatible* mapping of the ring $\mathbb{Z}_2$ into itself.

'Compatible' is an algebraic term. In cryptography they used to speak of '$T$-functions on $n$-bit words' instead of 'compatible mappings of the residue ring $\mathbb{Z}/2^n$ into itself'.

# Terminology notes

This is a univariate $T$-function $F$:

$$(\chi_0, \chi_1, \chi_2, \ldots) \stackrel{F}{\mapsto} (\psi_0(\chi_0); \psi_1(\chi_0, \chi_1); \psi_2(\chi_0, \chi_1, \chi_2); \ldots).$$

Here $\chi_j \in \{0, 1\}$, and each $\psi_j(\chi_0, \ldots, \chi_j)$ is a Boolean function in Boolean variables $\chi_0, \ldots, \chi_j$.

Thus, $F$ sends a number with the base-2 expansion

$$\chi_0 + \chi_1 \cdot 2 + \chi_2 \cdot 2^2 + \cdots$$

to the number with the base-2 expansion

$$\psi_0(\chi_0) + \psi_1(\chi_0, \chi_1) \cdot 2 + \psi_2(\chi_0, \chi_1, \chi_2) \cdot 2^2 + \cdots$$

# Yet another observation

We conclude:

- $T$-functions on $n$-bit words are just approximations of 2-adic compatible functions (i.e., functions that satisfy Lipschitz condition with coefficient 1) up to a precision $2^{-n}$ w. r. t. the 2-adic metric.

  That is, a $T$-function on $n$-bit words is just a reduction modulo $2^n$ of a 2-adic function that satisfy Lipschitz condition with coefficient 1

# Yet another observation

We conclude:

- $T$-functions on $n$-bit words are just approximations of 2-adic compatible functions
- To study properties of compatible functions (hence, properties of $T$-functions) one may use 2-adic analysis, since compatible functions are continuous.

# Yet another observation

We conclude:

- $T$-functions on $n$-bit words are just approximations of 2-adic compatible functions
- To study properties of compatible functions (hence, properties of $T$-functions) one may use 2-adic analysis
- In addition to the basic ship operations, to construct compatible functions one may use also subtraction, division by an odd integer, exponentiation of an odd integer

# Wild functions

For instance, a computer evaluates the following wild-looking function correctly, up to the best 2-adic precision he can achieve:

# Wild functions

For instance, a computer evaluates the following wild-looking function correctly, up to the best 2-adic precision he can achieve:

$$g(x) = \left(1 - 2 \cdot \frac{x \text{ AND } x^2 + x^3 \text{ OR } x^4}{3 - 4 \cdot (5 + 6x^5)^{x^6 \text{ XOR } x^7}}\right)^{7 - \frac{8x^8}{9 + 10x^9}}$$

# The virtual world is the non-Archimedean world!

The virtual world is the non-Archimedean world!

All triangles are isosceles!

The virtual world is the non-Archimedean world!

All triangles are isosceles!

Every point inside a circle is a center of the circle!

# Important:

There is a tight connection between the invertibility property/single cycle property of $T$-functions and metric properties of the corresponding 2-adic functions

# More 2-adic analysis

The space $\mathbb{Z}_2$ is a measurable space, which is endowed with a natural probabilistic measure, the *normalized Haar measure* $\mu_2$.

# More 2-adic analysis

The space $\mathbb{Z}_2$ is a measurable space, which is endowed with a natural probabilistic measure, the *normalized Haar measure* $\mu_2$. Namely, the set $a + 2^k\mathbb{Z}_2$, i.e., the set of all 2-adic integers that are congruent to $a$ modulo $2^k$, is a ball of radius $2^{-k}$. By the definition, the volume of this ball is $\mu_2(a + 2^k\mathbb{Z}_2) = 2^{-k}$.

# More 2-adic analysis

The space $\mathbb{Z}_2$ is a measurable space, which is endowed with a natural probabilistic measure, the *normalized Haar measure* $\mu_2$.
The mapping $F : \mathbb{S} \to \mathbb{S}$ of the measurable space $\mathbb{S}$ with a probabilistic measure $\mu$ is said to *preserve measure* $\mu$ (or *to be* $\mu$*-preserving*) iff $\mu(F^{-1}(S)) = \mu(S)$ for every measurable subset $S \subset \mathbb{S}$.

# More 2-adic analysis

The space $\mathbb{Z}_2$ is a measurable space, which is endowed with a natural probabilistic measure, the *normalized Haar measure* $\mu_2$.
The mapping $F : \mathbb{S} \to \mathbb{S}$ of the measurable space $\mathbb{S}$ with a probabilistic measure $\mu$ is said to *preserve measure* $\mu$ (or *to be* $\mu$*-preserving*) iff $\mu(F^{-1}(S)) = \mu(S)$ for every measurable subset $S \subset \mathbb{S}$. A $\mu$-preserving mapping $F$ is said to be *ergodic* iff $\mu(S) = 1$ or $\mu(s) = 0$ for every measurable $S$ such that $F^{-1}(S) \subset S$.

# More 2-adic analysis

The space $\mathbb{Z}_2$ is a measurable space, which is endowed with a natural probabilistic measure, the *normalized Haar measure* $\mu_2$.
The mapping $F : \mathbb{S} \to \mathbb{S}$ of the measurable space $\mathbb{S}$ with a probabilistic measure $\mu$ is said to *preserve measure* $\mu$ (or *to be* $\mu$*-preserving*) iff $\mu(F^{-1}(S)) = \mu(S)$ for every measurable subset $S \subset \mathbb{S}$. A $\mu$-preserving mapping $F$ is said to be *ergodic* iff $\mu(S) = 1$ or $\mu(s) = 0$ for every measurable $S$ such that $F^{-1}(S) \subset S$. Loosely speaking, the invariant set of the ergodic mapping is either nothing, or everything.

# Using 2-adic analysis

A compatible mapping $F : \mathbb{Z}_2 \to \mathbb{Z}_2$ is said to be *bijective (resp., transitive) modulo* $2^k$ iff the induced mapping $x \mapsto F(x) \pmod{2^k}$ is a permutation (resp., a permutation with a single cycle) on $\mathbb{Z}/2^k$.

# Using 2-adic analysis

A compatible mapping $F:\mathbb{Z}_2\to\mathbb{Z}_2$ is said to be *bijective (resp., transitive) modulo* $2^k$ iff the induced mapping $x\mapsto F(x)\pmod{2^k}$ is a permutation (resp., a permutation with a single cycle) on $\mathbb{Z}/2^k$.

**Theorem 1.** (Anashin, 2002) *A compatible mapping* $F:\mathbb{Z}_2\to\mathbb{Z}_2$ *is bijective* (*accordingly, transitive* ) *modulo* $2^k$ *for all* $k=1,2,3,\ldots$ *iff it is measure-preserving* (*or, accordingly, ergodic* ) *with respect to the normalized Haar measure* $\mu_2$ *on* $\mathbb{Z}_2$

# Using 2-adic analysis

A compatible mapping $F : \mathbb{Z}_2 \to \mathbb{Z}_2$ is said to be *bijective (resp., transitive) modulo* $2^k$ iff the induced mapping $x \mapsto F(x) \pmod{2^k}$ is a permutation (resp., a permutation with a single cycle) on $\mathbb{Z}/2^k$.

- measure preservation=invertibility $\pmod{2^k}$ for all $k \in \mathbb{N}$
- ergodicity=single cycle property $\pmod{2^k}$ for all $k \in \mathbb{N}$

## Important:

Thus, ergodic functions could serve as state update functions, whereas measure preserving functions could serve as output functions of the PRNG.

# Important:

We must know how to construct ergodic/measure-preserving functions out of basic chip instructions

# Using 2-adic analysis once again

To construct measure-preserving/ergodic functions, very often we could use the following effect, which is due to the '2-adic smoothness' of compatible functions:

A compatible function $F : \mathbb{Z}_2 \to \mathbb{Z}_2$ is measure-preserving/ergodic iff the corresponding $T$-function $F \pmod{2^n}$ on $n$-bit words (which is merely an approximation of $F$ with precision $\frac{1}{2^n}$) is invertible/with a single cycle property!

# Using 2-adic analysis once again

For crypto matters this gives:

To verify whether a $T$-function is invertible/with a single cycle property on $N$-bit words (where $N$ is big) one should check whether it is invertible/with a single cycle property on $n$-bit words, where $n$ is often rather small!

# Using 2-adic derivations

**Theorem 2.** (Anashin, 1993) Let a compatible function $F : \mathbb{Z}_2 \to \mathbb{Z}_2$ be uniformly differentiable on $\mathbb{Z}_2$. Then $F$ is ergodic if and only if it is transitive modulo $2^{N_2(F)+2}$ Here $N_2(F)$ is such that

$$\left\| \frac{F(x+h) - F(x)}{h} - F'(x) \right\|_2 \leq \frac{1}{4}$$

whenever $\|h\|_2 \leq 2^{-N_2(F)}$.

# Using 2-adic derivations

**Theorem 2.** (Anashin, 1993) Let a compatible function $F : \mathbb{Z}_2 \to \mathbb{Z}_2$ be uniformly differentiable on $\mathbb{Z}_2$. Then $F$ is ergodic if and only if it is transitive modulo $2^{N_2(F)+2}$

*Example.* (Klimov and Shamir, 2002) *The function* $x + (x^2 \,\mathsf{OR}\, 5)$ *is ergodic.*

# Using 2-adic derivations

**Theorem 2.** (Anashin, 1993) Let a compatible function $F : \mathbb{Z}_2 \to \mathbb{Z}_2$ be uniformly differentiable on $\mathbb{Z}_2$. Then $F$ is ergodic if and only if it is transitive modulo $2^{N_2(F)+2}$

*Example.* (Klimov and Shamir, 2002) *The function* $x + (x^2 \,\mathsf{OR}\, 5)$ *is ergodic.*

Note: In their publication Klimov and Shamir write that "...neither the invertibility nor the cycle structure of $x + (x^2 \,\mathsf{OR}\, 5)$ could be determined by his (*i.e., Anashin's*) techniques." Quite the opposite, this could be easily determined by these techniques:

# Using 2-adic derivations

**Theorem 2.** (Anashin, 1993) Let a compatible function $F : \mathbb{Z}_2 \to \mathbb{Z}_2$ be uniformly differentiable on $\mathbb{Z}_2$. Then $F$ is ergodic if and only if it is transitive modulo $2^{N_2(F)+2}$

*Example.* (Klimov and Shamir, 2002) *The function* $x + (x^2 \,\mathrm{OR}\, 5)$ *is ergodic.*

*Proof.* The function $F(x) = x + (x^2 \,\mathrm{OR}\, 5)$ is uniformly differentiable on $\mathbb{Z}_2$:
$F'(x) = 1 + 2x \cdot (x \,\mathrm{OR}\, 5)' = 1 + 2x$, and $N_2(F) = 3$ since obviously $(x + h) \,\mathrm{OR}\, 5 = (x \,\mathrm{OR}\, 5) + h$ whenever $h \equiv 0 \pmod 8$. Now to prove that $F$ is ergodic, in view of the above theorem it suffices to demonstrate that $F$ induces a permutation with a single cycle on $\mathbb{Z}/32$. One verifies this by direct calculations. □

# How to determine ergodic functions?

The following results, as well as the preceding ones, remain true (with some minor exceptions) for arbitrary prime $p$.
Any function $F: \mathbb{Z}_p \to \mathbb{Z}_p$ could be represented by *Mahler's interpolation series*: $F(x) = \sum_{j=0}^{\infty} c_j \binom{x}{j}$ for suitable $c_j \in \mathbb{Z}_p$. Recall

$$\binom{x}{i} = \begin{cases} \dfrac{x(x-1)\cdots(x-i+1)}{i!}, & \text{for } i = 1, 2, \ldots; \\ 1, & \text{for } i = 0. \end{cases}$$

An attempt to find an answer in terms of Mahler's interpolation series looks quite natural!

# How to determine ergodic functions?

**Theorem 3.** (Anashin, 1993) *For $p = 2$ the function $F : \mathbb{Z}_p \to \mathbb{Z}_p$ is compatible and ergodic iff*

$$F(x) = 1 + x + \sum_{i=1}^{\infty} c_i \cdot p^{\lfloor \log_p(i+1) \rfloor + 1} \binom{x}{i},$$

*for suitable $c_i \in \mathbb{Z}_p$.* (Note: For $p \neq 2$ remain sufficient, and not necessary).

# Examples

For $p = 2$ the following is true:

- (Larin, early 80th; published 2002) A polynomial with integer coefficients is ergodic iff it is transitive modulo 8.

# Examples

For $p = 2$ the following is true:

- (Anashin, 1993) The function $F(x) = a_0 + b_1 \cdot (x \oplus a_1) + b_2 \cdot (x \oplus a_2) + \cdots$ is ergodic iff it is transitive modulo $4$.

# Examples

For $p = 2$ the following is true:

- (Anashin, 1993) The function $F(x) = a + a_0 \cdot \delta_0(x) + a_1 \cdot \delta_1(x) + \cdots$ is compatible and ergodic iff $\|a\|_2 = 1$, $a_0 \equiv 1 \pmod 4$, and $\|a_j\|_2 = 1$ for $j = 1, 2, \ldots$ Here, we recall, $\delta_j(x) = \frac{1}{2^j}(x \text{ AND } 2^j)$ is the $j$-th bit in the base-2 expansion of $x$ (we start enumeration with $j = 0$). In other words, a compatible function $b + b_0 \cdot (x \text{ AND } 1) + b_1 \cdot (x \text{ AND } 2) + b_2 \cdot (x \text{ AND } 2^2) + \cdots$ is ergodic iff $b \equiv 1 \pmod 2$, $b_0 \equiv 1 \pmod 4$, and $b_j \equiv 1 \pmod 2$ for all $j = 1, 2, 3, \ldots$.

# Examples

For $p = 2$ the following is true:

- (Anashin, 1993) For arbitrary polynomials $u(x), v(x) \in \mathbb{Z}_2[x]$ the entire function

$$F(x) = \frac{v(x)}{2 \cdot u(x) + 1}$$

is ergodic iff it is transitive modulo 8

# Examples

For $p = 2$ the following is true:

- (Kotomina, 1999) The function

$$f(x) = (\dots((((x+c_0)\oplus d_0)+c_1)\oplus d_1)+\cdots+c_m)\oplus d_m,$$

is ergodic iff $f$ is transitive modulo 4

# Examples

For $p = 2$ the following is true:

- (Anashin, 2002) The function $F(x) = a \cdot x + a^x$ is ergodic iff $a$ is odd

# Examples

For $p = 2$ the following is true:

- (Anashin, 2002) A polynomial $f(x) \in \mathbb{Q}[x]$ of degree $d$ with rational (and not necessarily integral) coefficients is integer-valued (i.e., $f(\mathbb{Z}_2) \subset \mathbb{Z}_2$)) compatible, and ergodic iff $f$ takes integral values at the points $0, 1, \ldots, 2^{\lfloor \log_2(\deg f) \rfloor + 3} - 1$, and the mapping $z \mapsto f(z) \bmod 2^{\lfloor \log_2(\deg f) \rfloor + 3}$, is compatible and transitive on $\mathbb{Z}/2^{\lfloor \log_2 d \rfloor + 3}$ (i.e., modulo the biggest power of 2 not exceeding $8d$); i.e., to verify whether all three properties hold simultaneously, one has to make approximately $8d$ evaluations of $f(x)$

# Explicit formulae

The following theorem gives a general construction that enables one to build all ergodic mappings out of compatible ones.

**Theorem 4.** (Anashin, 2002) *Denote $\Delta U(x) = U(x+1) - U(x)$. For $p = 2$ the function $F : \mathbb{Z}_p \to \mathbb{Z}_p$ is compatible and ergodic $\Leftrightarrow$ $F(x) = 1 + x + p \cdot \Delta U(x)$, where $U : \mathbb{Z}_p \to \mathbb{Z}_p$ is arbitrary compatible function.*

*Note.* For $p \neq 2$ only $\Leftarrow$ is true.

*Note.* Recall that any composition of basic microchip operations (without cyclic shifts, and shifts towards less significant bits) is a compatible function on $\mathbb{Z}_2$!

# Usage

The presented results concern non-Archimedean autonomous dynamical systems on $\mathbb{Z}_2$, which are *not* chaotic, only ergodic.

In fact, these systems are *not* sensitive to minor perturbations of the initial position; moreover, they are isometries of the space $\mathbb{Z}_2$.

# Usage

The presented results concern non-Archimedean autonomous dynamical systems on $\mathbb{Z}_2$, which are *not* chaotic, only ergodic.

Yet these dynamical systems are good for state update functions of the PRNG, since they satisfy the conditions we mentioned before.

# Usage

The presented results concern non-Archimedean autonomous dynamical systems on $\mathbb{Z}_2$, which are *not* chaotic, only ergodic.

Yet these dynamical systems are good for state update functions of the PRNG, since they satisfy the conditions we mentioned before. Moreover, with the use of state update functions of this kind one could design flexible PRNG's, where not only the initial state, but also the state update function depends on key.

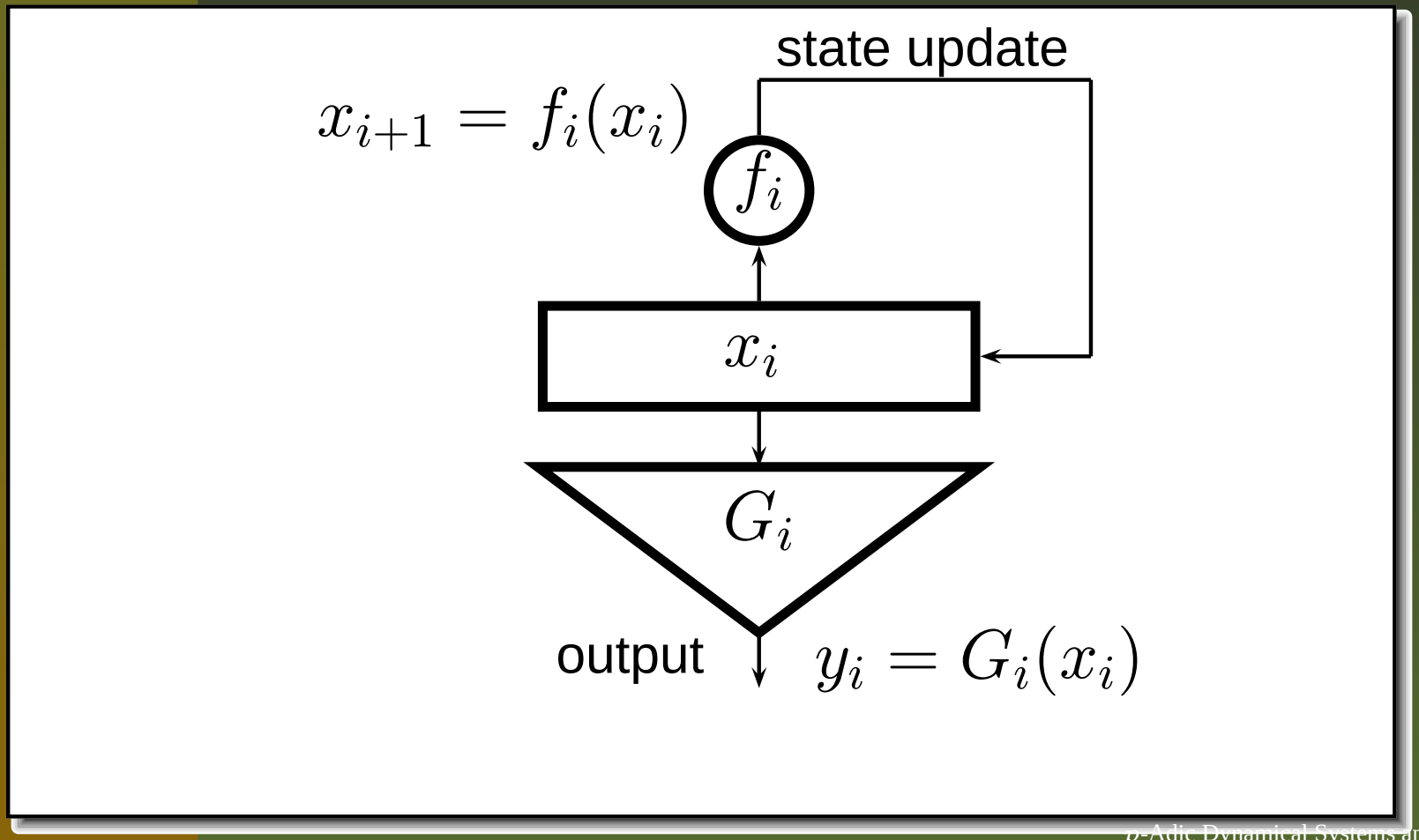
# Usage

The presented results concern non-Archimedean autonomous dynamical systems on $\mathbb{Z}_2$, which are *not* chaotic, only ergodic.

Yet these dynamical systems are good for state update functions of the PRNG, since they satisfy the conditions we mentioned before. Similar theory is developed for measure-preserving mappings, which are good for output functions (we have to omit details due to the time constraints). In whole, these ideas lead to fast and flexible stream ciphers.

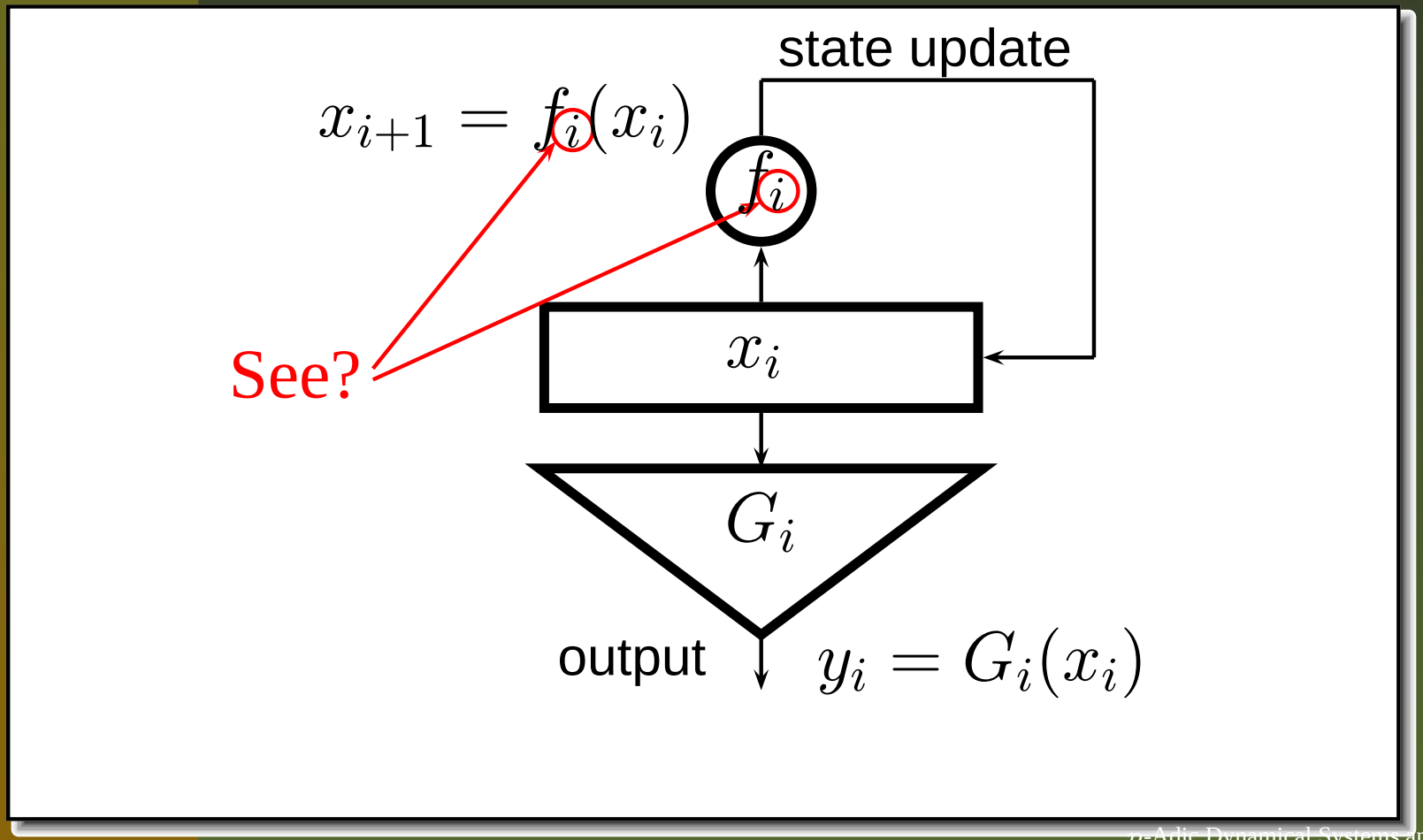
# Adding flexibility... and security

A *counter-dependent PRNG* also produces pseudorandom sequences. See the difference with ordinary PRNG?



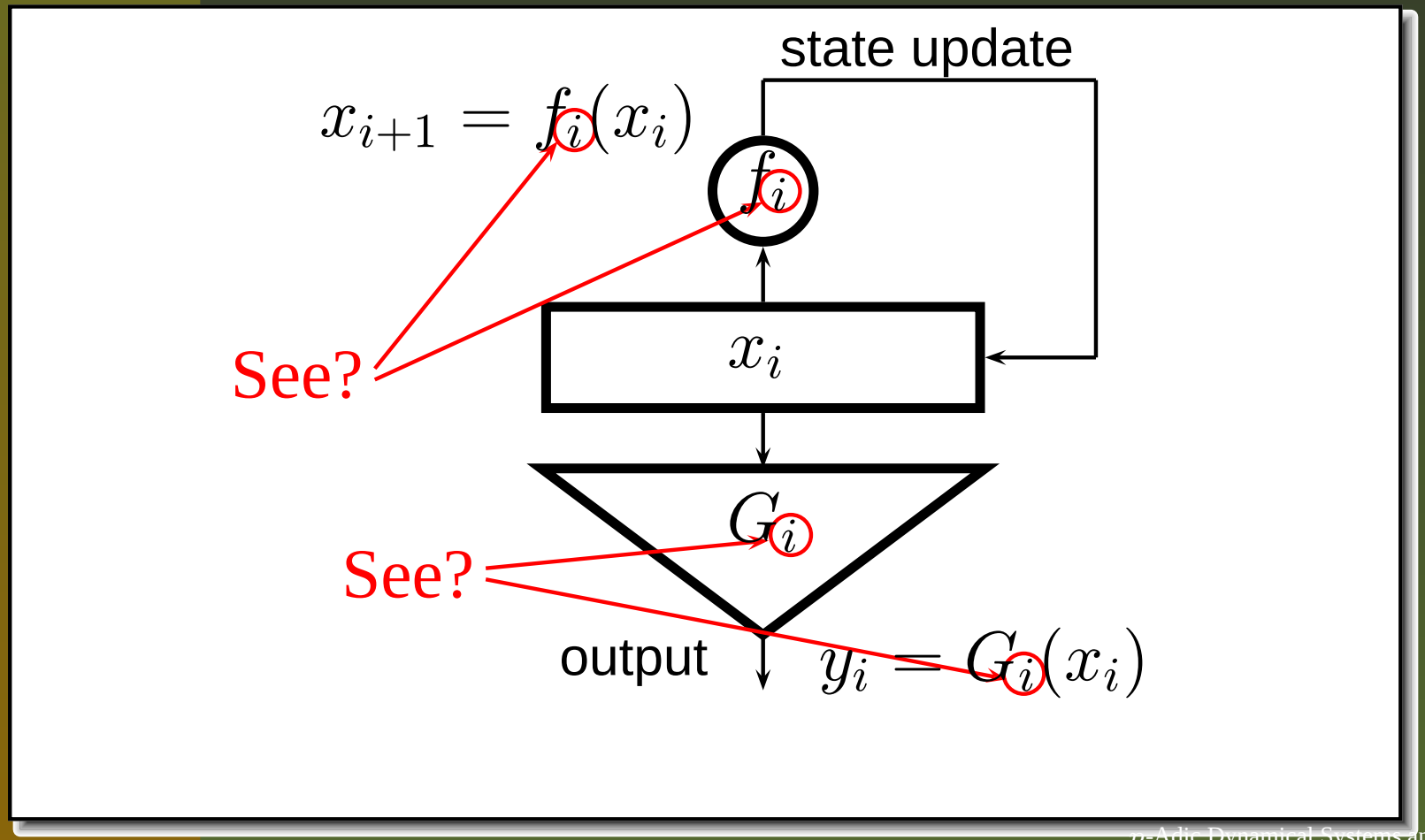
# Adding flexibility... and security

A *counter-dependent PRNG* also produces pseudorandom sequences. See the difference with ordinary PRNG?



# Adding flexibility... and security

A *counter-dependent PRNG* also produces pseudorandom sequences. See the difference with ordinary PRNG?



# Dynamical systems revisited

Recall that ordinary PRNG corresponds to an *autonomous* dynamical system.

# Dynamical systems revisited

Recall that ordinary PRNG corresponds to an *autonomous* dynamical system.

This is a *non-autonomous* dynamical system, which is a counterpart of a counter-dependent PRNG in dynamics.

A non-autonomous dynamical system is a dynamical system driven by another dynamical system.

# Dynamical systems revisited

Recall that ordinary PRNG corresponds to an *autonomous* dynamical system.

This is a *non-autonomous* dynamical system, which is a counterpart of a counter-dependent PRNG in dynamics.

A theory similar to that of the preceding is developed for counter-dependent PRNG.

# Dynamical systems revisited

Recall that ordinary PRNG corresponds to an *autonomous* dynamical system.

This is a *non-autonomous* dynamical system, which is a counterpart of a counter-dependent PRNG in dynamics.

A theory similar to that of the preceding is developed for counter-dependent PRNG. The main tool to construct a counter-dependent PRNG that outputs a sequence of a maximum period length, is a *skew shift*.

# Skew shifts, wreath products, etc.

What is a skew shift?

# Skew shifts, wreath products, etc.

What is a skew shift?

Given a mapping $U : Z \to Z$ , and a set of mappings $\mathcal{V} = \{(V_z : X \to X) : z \in Z\}$, a *skew shift* (or, a *skew product* or, a *wreath product*) is a mapping

$$U \leftthreetimes \mathcal{V} : (z, x) \mapsto (U(z), V_z(x))$$

of the Cartesian product $Z \times X$ into itself.

# Skew shifts, wreath products, etc.

What is a skew shift?

Given a mapping $U : Z \to Z$ , and a set of mappings $\mathcal{V} = \{(V_z : X \to X) : z \in Z\}$, a *skew shift* (or, a *skew product* or, a *wreath product*) is a mapping

$$U \leftthreetimes \mathcal{V} : (z, x) \mapsto (U(z), V_z(x))$$

of the Cartesian product $Z \times X$ into itself.

Obviously, the skew shift $U \leftthreetimes \mathcal{V}$ is bijective whenever both $U$ and all $V_z$ are bijective.

# Skew shifts, wreath products, etc.

What is a skew shift?

Given a mapping $U\colon Z \to Z$ , and a set of mappings $\mathcal{V} = \{(V_z\colon X \to X)\colon z \in Z\}$, a *skew shift* (or, a *skew product* or, a *wreath product*) is a mapping

$$U \leftthreetimes \mathcal{V}\colon (z,x) \mapsto (U(z), V_z(x))$$

of the Cartesian product $Z \times X$ into itself.

Skew shifts are familiar to crypto community; recall Feistel network: The mapping it is based on is a skew shift $(z,x) \mapsto (z, z \oplus f(x))$, where $z, x \in \mathbb{B}^n$, $f\colon \mathbb{B}^n \to \mathbb{B}^n$.

# Skew shifts, wreath products, etc.

What is a skew shift?

Given a mapping $U : Z \to Z$ , and a set of mappings $\mathcal{V} = \{(V_z : X \to X) : z \in Z\}$, a *skew shift* (or, a *skew product* or, a *wreath product*) is a mapping

$$U \leftthreetimes \mathcal{V} : (z, x) \mapsto (U(z), V_z(x))$$

of the Cartesian product $Z \times X$ into itself.

Skew shifts (in dynamical systems theory), which are also known under the name of wreath products (in group theory, in automata theory) are often used to obtain new objects with desirable properties out of given objects with known properties.

# Using the skew shifts

**Theorem 5.** (Anashin, 2004) *Let $\mathcal{F} = \{f_0, \ldots, f_{m-1}\}$ be a finite sequence of compatible measure preserving mappings of $\mathbb{Z}_2$ onto itself such that*

*(i) the sequence $\{(f_{i \bmod m}(0)) \bmod 2 \colon i = 0, 1, 2, \ldots\}$ is purely periodic, its shortest period is of length $m$;*

*(ii) $\sum_{i=0}^{m-1} f_i(0) \equiv 1 \pmod 2$;*

*(iii) $\sum_{j=0}^{m-1} \sum_{z=0}^{2^t-1} f_j(z) \equiv 2^t \pmod{2^{t+1}}$ for all $t = 1, 2, \ldots$.*

*Then the recurrence sequence $\mathcal{Z}$ defined by the relation $x_{i+1} = f_{i \bmod m}(x_i)$ is strictly uniformly distributed modulo $2^n$ for all $n = 1, 2, \ldots$ : That is, modulo each $2^n$ the sequence $\mathcal{Z}$ is purely periodic, its shortest period is of length $2^n m$, and each element of $\mathbb{Z}/2^n$ occurs at the period exactly $m$ times.*

# Using the skew shifts

*Example.* Given 2-adic numbers $c_0, \ldots, c_{m-1} \in \mathbb{Z}_2$, $m > 1$ odd, and compatible ergodic mappings (=$T$-functions with a single cycle property) $h_0, \ldots, h_{m-1}$. (The latter either could be stored in memory, or could be produced on-fly out of basic chip instructions, see e.g. theorem 4) The sequence $\{x_{i+1} = f_{i \bmod m}(x_i)\}$ of internal states of a counter-dependent PRNG is periodic modulo $2^n$ and strictly uniformly distributed modulo $2^n$ (that is, each $a \in \mathbb{Z}/2^n$ occurs at the period the same number of times), and the length of its shortest period is $m \cdot 2^n$ (that is, maximum possible) , if

# Using the skew shifts

*Example.* Given 2-adic numbers $c_0, \ldots, c_{m-1} \in \mathbb{Z}_2$, $m > 1$ odd, and compatible ergodic mappings $h_0, \ldots, h_{m-1}$. The sequence of internal states of a counter-dependent PRNG is periodic modulo $2^n$ and strictly uniformly distributed modulo $2^n$ , and the length of its shortest period is $m \cdot 2^n$ , if

- the sequence $\{c_{i \bmod m} \bmod 2 \colon i = 0, 1, 2, \ldots\}$ is periodic, and $m$ is the length of its shortest period

# Using the skew shifts

*Example.* Given 2-adic numbers $c_0, \ldots, c_{m-1} \in \mathbb{Z}_2$, $m > 1$ odd, and compatible ergodic mappings $h_0, \ldots, h_{m-1}$. The sequence of internal states of a counter-dependent PRNG is periodic modulo $2^n$ and strictly uniformly distributed modulo $2^n$ , and the length of its shortest period is $m \cdot 2^n$ , if

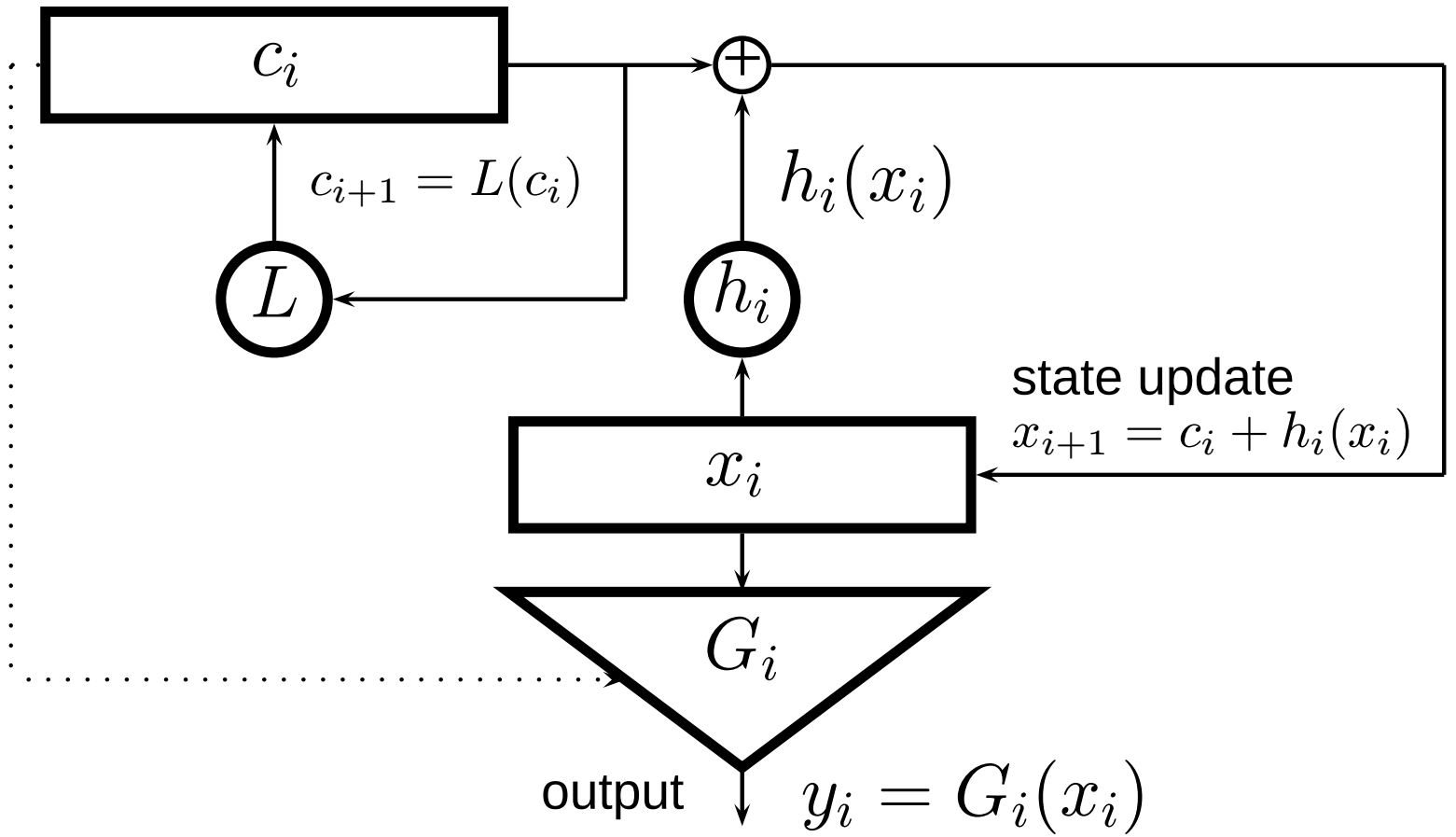
- the sequence $\{c_{i \bmod m} \bmod 2 \colon i = 0, 1, 2, \ldots\}$ is periodic, and $m$ is the length of its shortest period
- $\sum_{j=0}^{m-1} c_j \equiv 0 \pmod 2$

# Using the skew shifts

*Example.* Given 2-adic numbers $c_0, \ldots, c_{m-1} \in \mathbb{Z}_2$, $m > 1$ odd, and compatible ergodic mappings $h_0, \ldots, h_{m-1}$. The sequence of internal states of a counter-dependent PRNG is periodic modulo $2^n$ and strictly uniformly distributed modulo $2^n$ , and the length of its shortest period is $m \cdot 2^n$ , if

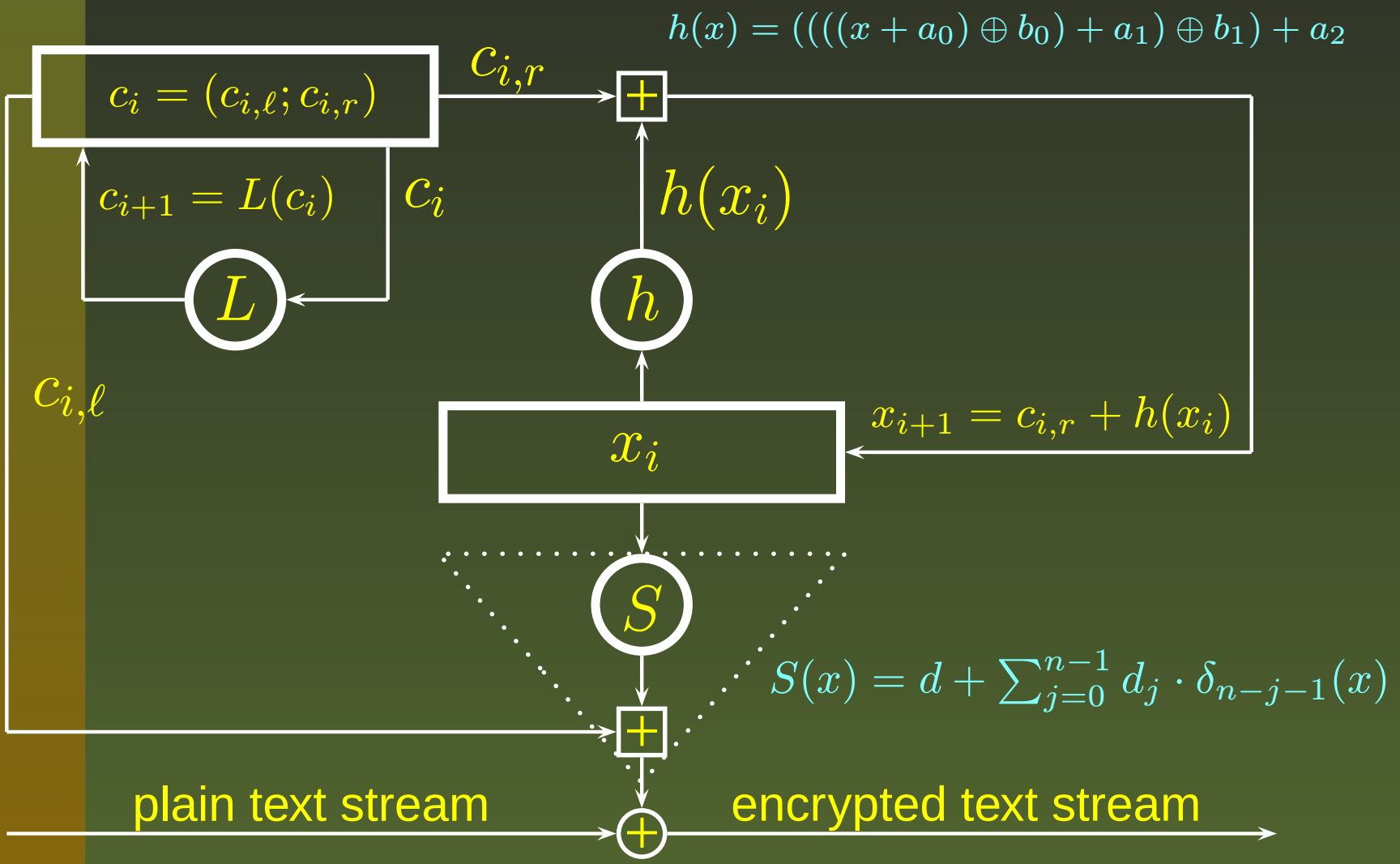
- the sequence $\{c_{i \bmod m} \bmod 2 \colon i = 0, 1, 2, \ldots\}$ is periodic, and $m$ is the length of its shortest period
- $\sum_{j=0}^{m-1} c_j \equiv 0 \pmod 2$
- $f_j(x) = c_j \oplus h_j(x)$, or $f_j(x) = c_j + h_j(x)$

# Example circuit

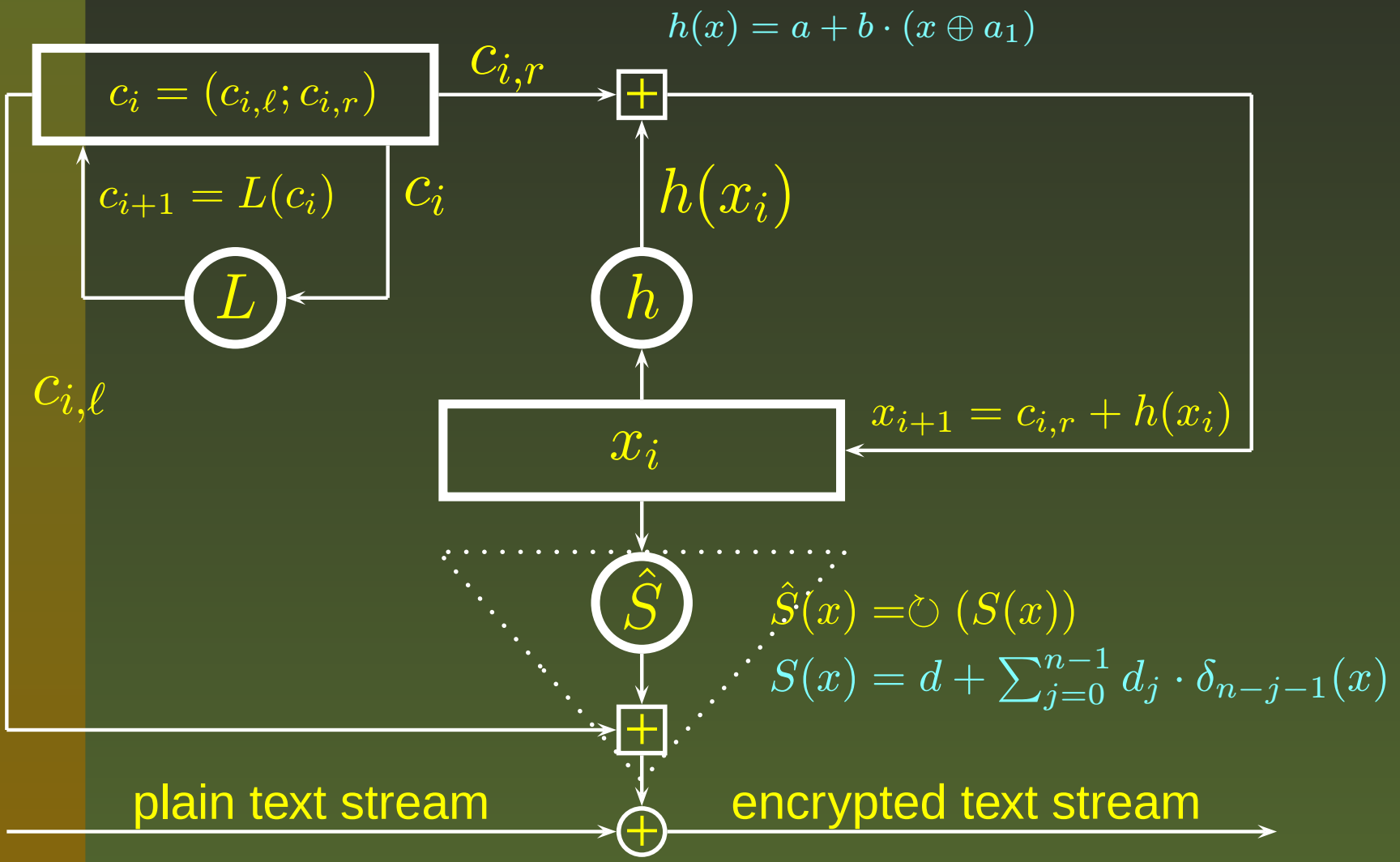



$L(c) = 2 \cdot c \oplus \overline{u} \cdot \delta_{n-1}(c)$; $\overline{u}$ agrees with coefficients of the polynomial $u$

# The ABC stream cipher: 6 Gbits/sec



# The ABC stream cipher: 6 Gbits/sec



# The ABC stream cipher: Properties.

The following is *proved*:

- Length $P$ of the period of the output sequence is $(2^{2n-1}-1)\cdot 2^n$

# The ABC stream cipher: Properties.

The following is *proved*:

- Length $P$ of the period of the output sequence is $(2^{2n-1}-1)\cdot 2^n$
- $n$-tuples of the output are uniformly distributed:

$$\left|\frac{\mu(a)}{P}-\frac{1}{2^n}\right|<\frac{1}{\sqrt{P}},$$

where $\mu(a)$ is the number of occurrences of an $n$-tuple $a\in\mathbb{Z}/2^n$ at the period.

Note: For a truly random sequence of $n$-bit words of length $P$ the above inequality holds with probability $>1-\frac{1}{2^n}$.

# The ABC stream cipher: Properties.

The following is *proved*:

- Length $P$ of the period of the output sequence is $(2^{2n-1}-1)\cdot 2^n$
- $n$-tuples of the output are uniformly distributed:

$$\left|\frac{\mu(a)}{P}-\frac{1}{2^n}\right|<\frac{1}{\sqrt{P}},$$

where $\mu(a)$ is the number of occurrences of an $n$-tuple $a\in\mathbb{Z}/2^n$ at the period.

- Linear complexity (over $\mathbb{Z}/2$) of the output sequence exceeds $2^{n-1}$

# How random is the output?

*Frequency tests* are those that consider occurrences of (overlapping) $\ell$-tuples in a binary output.

# How random is the output?

*Frequency tests* are those that consider occurrences of (overlapping) $\ell$-tuples in a binary output.
That is, given a sequence $\mathcal{X} = x_0, x_1, x_2, \ldots$ of non-negative rational integers, one represents $x_i \bmod 2^n$ as an $n$-bit word $\overline{x_i \bmod 2^n}$ (base-2 expansion of $x_i \bmod 2^n$), considers a concatenation

$$\mathcal{X}'_n = \overline{x_i \bmod 2^n}\ \overline{x_{i+1} \bmod 2^n}\ \overline{x_{i+2} \bmod 2^n} \ldots$$

and counts occurrences of patterns
$0, 1, 00, 01, 10, 11, 000, 001, \ldots$.

# How random is the output?

*Frequency tests* are those that consider occurrences of (overlapping) $\ell$-tuples in a binary output.
That is, given a sequence $\mathcal{X} = x_0, x_1, x_2, \ldots$ of non-negative rational integers, one represents $x_i \bmod 2^n$ as an $n$-bit word $\overline{x_i \bmod 2^n}$ (base-2 expansion of $x_i \bmod 2^n$), considers a concatenation

$$\mathcal{X}'_n = \overline{x_i \bmod 2^n}\ \overline{x_{i+1} \bmod 2^n}\ \overline{x_{i+2} \bmod 2^n} \ldots$$

and counts occurrences of patterns $0, 1, 00, 01, 10, 11, 000, 001, \ldots$. For a good sequence all the distributions must agree with the ones of a truly random sequence. Obviously, this never holds for a periodic sequence.

# Randomness by Knuth

Donald Knuth in his “The Art of Computer Programming" calls a finite binary sequence of length $T$ random, whenever it satisfies the following condition:

$$\left| \frac{\nu(\beta_0 \dots \beta_{\ell-1})}{T} - \frac{1}{2^\ell} \right| \leq \frac{1}{\sqrt{T}}$$

for all $0 < \ell \leq \log_2 T$, where $\nu(\beta_0 \dots \beta_{\ell-1})$ is the number of occurrences of the pattern $\beta_0 \dots \beta_{\ell-1}$ in the sequence.

# Randomness by Knuth

Donald Knuth in his “The Art of Computer Programming" calls a finite binary sequence of length $T$ random, whenever it satisfies the following condition:

$$\left|\frac{\nu(\beta_0 \dots \beta_{\ell-1})}{T} - \frac{1}{2^\ell}\right| \leq \frac{1}{\sqrt{T}}$$

for all $0 < \ell \leq \log_2 T$, where $\nu(\beta_0 \dots \beta_{\ell-1})$ is the number of occurrences of the pattern $\beta_0 \dots \beta_{\ell-1}$ in the sequence. So its is quite natural to say that a periodic sequence is *random in the sense of Knuth* iff its shortest period satisfy the above condition.

# Randomness by Knuth

Donald Knuth in his “The Art of Computer Programming" calls a finite binary sequence of length $T$ random, whenever it satisfies the following condition:

$$\left| \frac{\nu(\beta_0 \dots \beta_{\ell-1})}{T} - \frac{1}{2^\ell} \right| \leq \frac{1}{\sqrt{T}}$$

for all $0 < \ell \leq \log_2 T$, where $\nu(\beta_0 \dots \beta_{\ell-1})$ is the number of occurrences of the pattern $\beta_0 \dots \beta_{\ell-1}$ in the sequence. Note that uniform distribution of the sequence $\mathcal{X}$ does not imply $\mathcal{X}'_n$ is random in the sense of Knuth!

# Randomness by Knuth

The sequences produced by our generators of the (maximum possible) period length $T$ are random in the sense of Knuth:

$$\left| \frac{\nu(\beta_0 \dots \beta_{\ell-1})}{T} - \frac{1}{2^\ell} \right| \leq \frac{1}{\sqrt{T}}$$

for all $0 < \ell \leq \log_2 T$, where $\nu(\beta_0 \dots \beta_{\ell-1})$ is the number of occurrences of the pattern $\beta_0 \dots \beta_{\ell-1}$ in the sequence.

# Linear complexity

*Linearity tests* are those that consider linear dependencies in the sequence.

# Linear complexity

**Definition.** Let $\mathcal{Z} = \{z_i\}$ be a sequence over a ring $R$. The *linear complexity* $\lambda_R(\mathcal{Z})$ of $\mathcal{Z}$ over $R$ is the smallest $r \in \mathbb{N}_0$ such that there exist $c, c_0, c_1, \ldots, c_{r-1} \in R$ (not all equal to $0$) such that for all $i = 0, 1, 2, \ldots$ holds

$$c + \sum_{j=0}^{r-1} c_j \cdot z_{i+j} = 0.$$

# Linear complexity

**Definition.** Let $\mathcal{Z} = \{z_i\}$ be a sequence over a ring $R$. The *linear complexity* $\lambda_R(\mathcal{Z})$ of $\mathcal{Z}$ over $R$ is the smallest $r \in \mathbb{N}_0$ such that there exist $c, c_0, c_1, \ldots, c_{r-1} \in R$ (not all equal to $0$) such that for all $i = 0, 1, 2, \ldots$ holds

$$c + \sum_{j=0}^{r-1} c_j \cdot z_{i+j} = 0.$$

For instance, if $R = \mathbb{Z}/p^n$; then geometrically this equation means that all the points $\left(\frac{z_i}{p^n}, \frac{z_{i+1}}{p^n}, \ldots, \frac{z_{i+r-1}}{p^n}\right)$, $i = 0, 1, 2, \ldots$, of a unit $r$-dimensional Euclidean hypercube fall into parallel hyperplanes.

# Linear complexity

In fact, linearity tests turn out to be ones of the most effective.

For example, linear congruential generators $x_{i+1} = a + b \cdot x_i \pmod{2^n}$ do not pass these tests. Linear complexity over $\mathbb{Z}/2^n$ of linear congruential generators is 2; hence, distribution of pairs in produced sequences is rather poor:

All the points that correspond to pairs of consecutive numbers fall into a small number of parallel straight lines in a unit square.

# Linear complexity

All $T$-functions with a single cycle property produce uniformly distributed sequences. However, some of these $T$-functions produce bad sequences, which have a number of linear dependencies modulo $p^n$, and poor distribution of pairs

# Linear complexity

All $T$-functions with a single cycle property produce uniformly distributed sequences. However, some of these $T$-functions produce bad sequences, which have a number of linear dependencies modulo $p^n$, and poor distribution of pairs

*Example.* A $T$-function $x + x^2$ OR $C$ has a single cycle property whenever $C \equiv 5 \pmod 8$, or $C \equiv 7 \pmod 8$ (Klimov and Shamir, 2002)
However, the distribution of pairs of the sequence produced by this $T$-function varies from satisfactory (when there are few 1's in more significant bit positions) to poor (when there are more 1's).

# Linear complexity

All $T$-functions with a single cycle property produce uniformly distributed sequences. However, some of these $T$-functions produce bad sequences, which have a number of linear dependencies modulo $p^n$, and poor distribution of pairs

This is not easy to find a $T$-function that guarantees good distribution of pairs.

For instance, this problem is not completely solved even for quadratic generators with a single cycle property, despite a number of works in the area (see e.g. Emmerich, 1997; Eichenauer-Hermann, 1995-1997, et. al.).

# Linear complexity

All $T$-functions with a single cycle property produce uniformly distributed sequences. However, some of these $T$-functions produce bad sequences, which have a number of linear dependencies modulo $p^n$, and poor distribution of pairs

However, we can prove that with respect to the linear complexity over residue ring the sequence $\mathcal{X}_n = \{f^i(x_0) \bmod p^n\}$ over $\mathbb{Z}/p^n$, generated by compatible ergodic polynomial $f(x) \in \mathbb{Q}_p[x]$ of degree $\geq 2$, is 'asymptotically good'.

# Linear complexity

All $T$-functions with a single cycle property produce uniformly distributed sequences. However, some of these $T$-functions produce bad sequences, which have a number of linear dependencies modulo $p^n$, and poor distribution of pairs

However, we can prove that with respect to the linear complexity over residue ring the sequence $\mathcal{X}_n = \{f^i(x_0) \bmod p^n\}$ over $\mathbb{Z}/p^n$, generated by compatible ergodic polynomial $f(x) \in \mathbb{Q}_p[x]$ of degree $\geq 2$, is 'asymptotically good'.

**Theorem.** (Anashin, 2002) $\lim_{n\to\infty} \lambda_{\mathbb{Z}/p^n}(\mathcal{X}_n) = \infty$ .

*Moreover,* $\lambda_{\mathbb{Z}/p^n}(\mathcal{X}_n)$ *tends to* $\infty$ *not slower than* $\log n$.

# Coordinate sequences: Bad news

The drawback of the sequence produced by a $T$-function $F : \mathbb{Z}/2^k \to \mathbb{Z}/2^k$ with the single cycle property is that the less significant is the bit, the shorter is the period of the sequence it outputs; that is:
Despite the length of the period of the sequence

$$\mathcal{S} = \{z_0 = z, z_1 = F(z_0), z_2 = F(z_1), \ldots\}$$

of $k$-bit words is $2^k$, the length of the period of the $j^{\text{th}}$ bit sequence (which is called the $j^{\text{th}}$*coordinate sequence*)

$$\mathcal{S}_j = \{\delta_j(z_0), \delta_j(z_1), \delta_j(z_2), \ldots, \delta_j(z_{i+1}), \ldots\}$$

is only $2^{j+1}$, $(j = 0, 1, \ldots, k-1)$.

# Coordinate sequences: Bad news

**Proposition** (Anashin, 2004) *The $j^{th}$ coordinate sequence $\mathcal{S}_j$ is purely periodic, and $2^{j+1}$ is the length of its shortest period. The second half of the period is a bitwise negation of the first half, i.e., $\zeta_{i+2^j} \equiv \zeta_i + 1 \pmod 2$ for each $i = 0, 1, 2, \ldots$. The linear complexity $\lambda_2(\mathcal{S}_j)$ of $\mathcal{S}_j$ over $GF(2)$ is exactly $2^j + 1$.*

# Coordinate sequences: Bad news

**Proposition** (Anashin, 2004) *The $j^{th}$ coordinate sequence $\mathcal{S}_j$ is purely periodic, and $2^{j+1}$ is the length of its shortest period. The second half of the period is a bitwise negation of the first half, i.e., $\zeta_{i+2^j} \equiv \zeta_i + 1 \pmod 2$ for each $i = 0, 1, 2, \ldots$. The linear complexity $\lambda_2(\mathcal{S}_j)$ of $\mathcal{S}_j$ over $GF(2)$ is exactly $2^j + 1$.*

*Note.* In fact, somewhat similar estimates hold for a 2-adic span, another measure of complexity of sequences, introduced by Klapper and Goresky. Similar results are true for coordinate sequences of the sequence of states of a counter-dependent PRNG.

# Coordinate sequences: Bad news

**Proposition** (Anashin, 2004) *The $j^{th}$ coordinate sequence $\mathcal{S}_j$ is purely periodic, and $2^{j+1}$ is the length of its shortest period. The second half of the period is a bitwise negation of the first half, i.e., $\zeta_{i+2^j} \equiv \zeta_i + 1 \pmod 2$ for each $i = 0, 1, 2, \ldots$. The linear complexity $\lambda_2(\mathcal{S}_j)$ of $\mathcal{S}_j$ over $GF(2)$ is exactly $2^j + 1$.*

Note that the expectation of the linear complexity $\lambda_2(\mathcal{C})$ of a random sequence $\mathcal{C}$ of length $T$ is $\frac{T}{2}$. Thus, the coordinate sequences are rather good with respect to their linear complexities.

# Coordinate sequences: Bad news

**Proposition** (Anashin, 2004) *The $j^{th}$ coordinate sequence $\mathcal{S}_j$ is purely periodic, and $2^{j+1}$ is the length of its shortest period. The second half of the period is a bitwise negation of the first half, i.e., $\zeta_{i+2^j} \equiv \zeta_i + 1 \pmod 2$ for each $i = 0, 1, 2, \ldots$. The linear complexity $\lambda_2(\mathcal{S}_j)$ of $\mathcal{S}_j$ over $GF(2)$ is exactly $2^j + 1$.*

However, from the proof of the proposition it follows that these good estimates holds only because the second half of the period of a coordinate sequence is a bitwise negation of the first half. In other words, the coordinate sequence is as 'complex' as the first half of its period

# Coordinate sequences: Bad news

**Proposition** (Anashin, 2004) *The $j^{th}$ coordinate sequence $\mathcal{S}_j$ is purely periodic, and $2^{j+1}$ is the length of its shortest period. The second half of the period is a bitwise negation of the first half, i.e., $\zeta_{i+2^j} \equiv \zeta_i + 1 \pmod 2$ for each $i = 0, 1, 2, \ldots$. The linear complexity $\lambda_2(\mathcal{S}_j)$ of $\mathcal{S}_j$ over $GF(2)$ is exactly $2^j + 1$.*
The important question is:

Given a $T$-function with a single cycle property, what bit sequence of length $2^j$ could be outputted as the first half of the period of the $j^{\text{th}}$ coordinate sequence?

# Coordinate sequences: Bad news

**Proposition** (Anashin, 2004) *The $j^{th}$ coordinate sequence $\mathcal{S}_j$ is purely periodic, and $2^{j+1}$ is the length of its shortest period. The second half of the period is a bitwise negation of the first half, i.e., $\zeta_{i+2^j} \equiv \zeta_i + 1 \pmod 2$ for each $i = 0, 1, 2, \ldots$. The linear complexity $\lambda_2(\mathcal{S}_j)$ of $\mathcal{S}_j$ over $GF(2)$ is exactly $2^j + 1$.*
The important question is:

Given a $T$-function with a single cycle property, what bit sequence of length $2^j$ could be outputted as the first half of the period of the $j^{\text{th}}$ coordinate sequence?

The answer is: ANY ONE, and independently of other coordinate sequences.

# Coordinate sequences: Good news

Let $\gamma_j(F,z) \in \mathbb{N}_0$ be such a number that its base-2 expansion agrees with the first half of the period of the $j^{\text{th}}$ coordinate sequence produced by the $T$-function $F$ with a single cycle property starting with the initial state $z$; that is,

$$\gamma_j(F,z) = \delta_j(F^{(0)}(z)) + 2\delta_j(F^{(1)}(z)) + \cdots + 2^{2^j-1}\delta_j(F^{(2^j-1)}(z)).$$

Obviously, $0 \le \gamma_j(F,z) \le 2^{2^j} - 1$.

**Theorem** (Anashin, 2004) *Let* $\Gamma = \{\gamma_j \in \mathbb{N}_0 : j = 0,1,2,\ldots\}$ *be an arbitrary sequence of non-negative rational integers such that* $0 \le \gamma_j \le 2^{2^j} - 1$ *for* $j = 0,1,2,\ldots$. *There exists a compatible and ergodic mapping* $F : \mathbb{Z}_2 \to \mathbb{Z}_2$ *and a 2-adic integer* $z \in \mathbb{Z}_2$ *such that*

$$\gamma_j \equiv \gamma_j(F,z) \pmod{2^{2^j}} \qquad (j = 0,1,2,\ldots)$$

# Coordinate sequences: Good news

**Theorem** (Anashin, 2004) *Let* $\Gamma = \{\gamma_j \in \mathbb{N}_0 : j = 0, 1, 2, \ldots\}$ *be an arbitrary sequence of non-negative rational integers such that* $0 \leq \gamma_j \leq 2^{2^j} - 1$ *for* $j = 0, 1, 2, \ldots$. *There exists a compatible and ergodic mapping* $F : \mathbb{Z}_2 \to \mathbb{Z}_2$ *and a 2-adic integer* $z \in \mathbb{Z}_2$ *such that*

$$\gamma_j \equiv \gamma_j(F, z) \pmod{2^{2^j}} \qquad (j = 0, 1, 2, \ldots)$$

Note: A proof of this theorem also uses p-adic techniques.
Note: A similar theorem holds for coordinate sequences of state sequences of counter-dependent PRNG of a maximum period length.

# Coordinate sequences: A remedy

What output function $G$ one should use? $G$ must add security, $G$ must be balanced (for not to spoil the uniform distribution), and $G$ must cure the very unpleasant 'low order bits effect' of $T$-functions.

# Coordinate sequences: A remedy

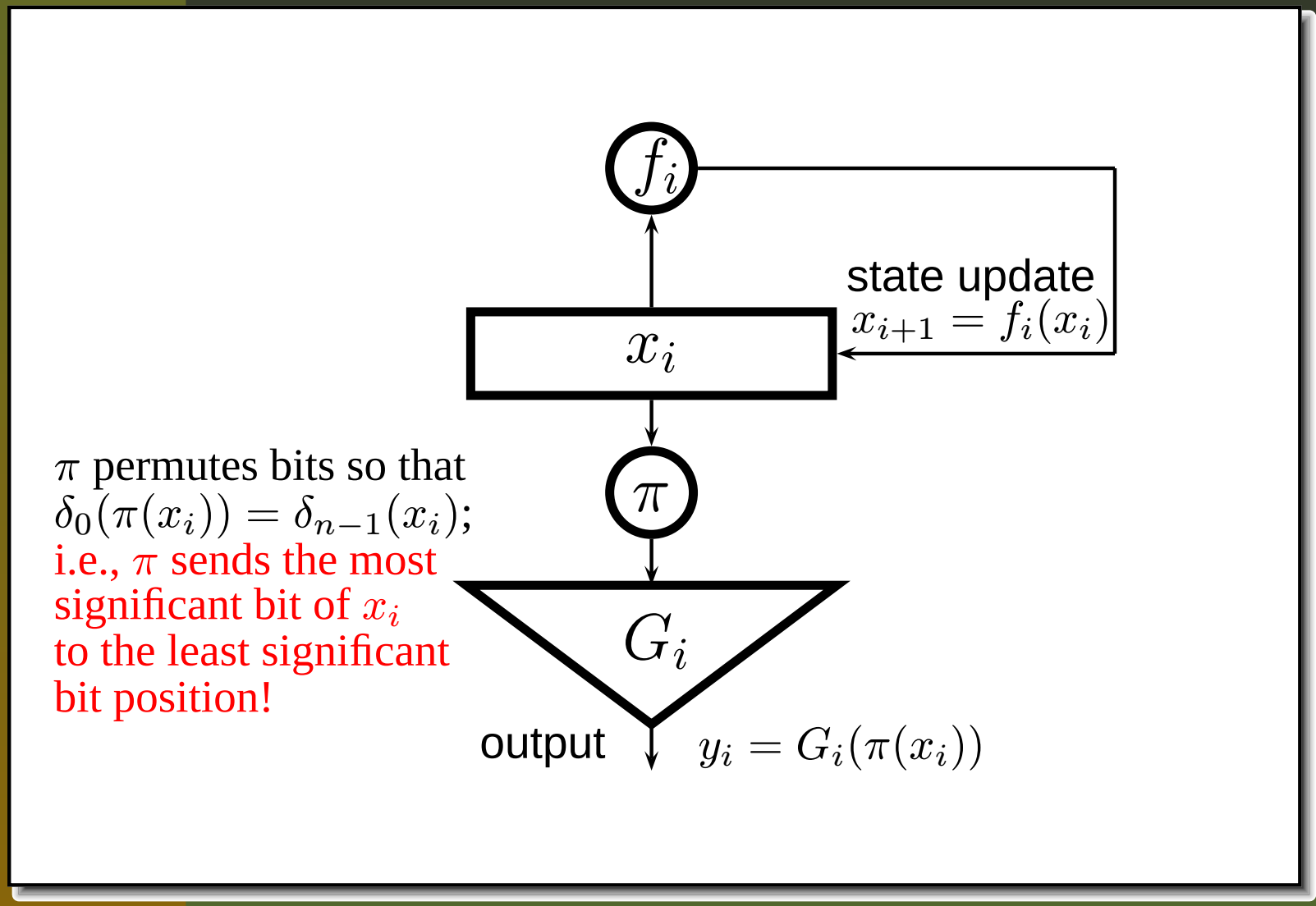
What output function $G$ one should use? $G$ must add security, $G$ must be balanced (for not to spoil the uniform distribution), and $G$ must cure the very unpleasant 'low order bits effect' of $T$-functions. One way (that might be good) is to truncate low order bits. Are there other ways?

# Coordinate sequences: A remedy

What output function $G$ one should use? $G$ must add security, $G$ must be balanced (for not to spoil the uniform distribution), and $G$ must cure the very unpleasant 'low order bits effect' of $T$-functions.
Since the 'low order bits effect' is an inherent property of $T$-functions, one should include in $G$ some basic chip operations other than $T$-functions. Thus, $G$ will not be a $T$-function any more. Could one construct $G$ this way, yet not 'spoil' good properties of the sequence of states?

# Coordinate sequences: A remedy

YES! This is how the solution looks schematically:




$\pi$ permutes bits so that $\delta_0(\pi(x_i)) = \delta_{n-1}(x_i)$; i.e., $\pi$ sends the most significant bit of $x_i$ to the least significant bit position!

# Coordinate sequences: A remedy

And this is how all this sounds mathematically:

**Proposition 1.** (Anashin, 2004) *Let* $G_i : \mathbb{Z}_2 \to \mathbb{Z}_2$ $(i = 0, 1, 2, \ldots, m-1)$ *be compatible and ergodic mappings* (=$T$-functions with a single cycle property). *For* $x \in \{0, 1, \ldots, 2^n - 1\}$ *let* $H_i(x) = (G_i(\pi(x))) \bmod 2^n$, *where* $\pi$ *is a permutation of bits of* $x \in \mathbb{Z}/2^n$ *such that* $\delta_0(\pi(x)) = \delta_{n-1}(x)$. *Consider a sequence* $\mathcal{H} = \{H_i(x_i)\}$, *where* $\{x_i\}$ *is the state update sequence of our counter-dependent PRNG (see e.g. the example circuit).Then the shortest period of the* $j^{th}$ *coordinate sequence* $\mathcal{H}_j = \delta_j(\mathcal{H})$ $(j = 0, 1, 2, \ldots, n-1)$ *is of length* $2^n k_j$ *for a suitable* $1 \le k_j \le m$. *Moreover, linear complexity of the sequence* $\mathcal{H}_j$ *exceeds* $2^{n-1}$, $\lambda_2(\mathcal{H}_j) > 2^{n-1}$.

# Three tools

Techniques that enable one to construct single cycle (resp., invertible, balanced) mappings out of basic chip operations mainly utilize the following three approaches:

- non-Archimedean ($p$-adic) analysis for $p = 2$;
- skew shifts (=wreath products);
- Boolean representations

We already have discussed the first and the second of these approaches.

# Boolean representations

The third of the three approaches, which is based on the theory of Boolean functions, is more straightforward, and could be applied directly only to relatively short and simple compositions of the basic chip instructions.

However, on the one hand, this approach is tightly connected with the skew shift techniques and, on the other hand, it lies in a background of some results obtained within the non-Archimedean approach.

# Boolean representations

By the definition, a univariate $T$-function $F$ is the mapping

$$(\chi_0, \chi_1, \chi_2, \ldots) \stackrel{F}{\mapsto} (\psi_0(\chi_0); \psi_1(\chi_0, \chi_1); \psi_2(\chi_0, \chi_1, \chi_2); \ldots),$$

where $\chi_j \in \{0, 1\}$, and each $\psi_j(\chi_0, \ldots, \chi_j)$ is a Boolean function in Boolean variables $\chi_0, \ldots, \chi_j$.

It turns out that one could determine whether $F$ is invertible/with a single cycle property by analyzing algebraic normal forms of Boolean functions $\psi_j$.

# Boolean representations: ANF

Recall that the *algebraic normal form*, ANF, of the Boolean function $\psi_j(\chi_0, \ldots, \chi_j)$ is the representation of this function via $\oplus$ (addition modulo 2= logical 'exclusive or') and $\cdot$ (multiplication modulo 2 =logical 'and'= conjunction).

# Boolean representations: ANF

In other words, the ANF of the Boolean function $\psi$ is its representation in the form

$$\psi(\chi_0, \ldots, \chi_j) = \beta \oplus \beta_0\chi_0 \oplus \beta_1\chi_1 \oplus \ldots \oplus \beta_{0,1}\chi_0\chi_1 \oplus \ldots,$$

where $\beta, \beta_0, \ldots \in \{0, 1\}$.

Recall that the *weight* of the Boolean function $\psi_j$ in $(j+1)$ variables is the number of $(j+1)$-bit words that *satisfy* $\psi_j$; that is, weight is the cardinality of the truth set of $\psi_j$.

# A folklore

**Theorem 6.** (folklore, more than 30 years old.) *A univariate $T$-function $F$*

$$(\chi_0, \chi_1, \chi_2, \ldots) \stackrel{F}{\mapsto} (\psi_0(\chi_0); \psi_1(\chi_0, \chi_1); \psi_2(\chi_0, \chi_1, \chi_2); \ldots),$$

*is invertible iff for each $j = 0, 1, \ldots$ the Boolean function $\psi_j$ in Boolean variables $\chi_0, \ldots, \chi_j$ is linear with respect to the variable $\chi_j$; that is, $F$ is invertible $\Leftrightarrow$ the ANF of each $\psi_j$ is of the form*

$$\psi_j(\chi_0, \ldots, \chi_j) = \chi_j \oplus \varphi_j(\chi_0, \ldots, \chi_{j-1}),$$

*where $\varphi_j$ is the Boolean function that does not depend on the variable $\chi_j$.*

# A folklore

**Theorem 7.** (folklore, more than 30 years old.) *The mapping $F$ has a single cycle property iff, additionally, the Boolean function $\varphi_j$ is of odd weight. The latter takes place if and only if $\varphi_0 = 1$, and the full degree of the Boolean function $\varphi_j$ for $j \geq 1$ is exactly $j$, that is, the ANF of $\varphi_j$ contains a monomial $\chi_0 \cdots \chi_{j-1}$.*

*Thus, $F$ has a single cycle property $\Leftrightarrow \psi_0(\chi_0) = \chi_0 \oplus 1$, and for $j \geq 1$ the ANF of each $\psi_j$ is of the form*

$$\psi_j(\chi_0, \ldots, \chi_j) = \chi_j \oplus \chi_0 \cdots \chi_{j-1} \oplus \theta_j(\chi_0, \ldots, \chi_{j-1}),$$

*where the weight of $\theta_j$ is even; i.e., $\deg \theta_j \leq j - 1$.*

# $T$-functions are also skew shifts!

*Note*: Theorem 5 is a generalization of these folklore theorems; the latter are special case of theorem 5 for $m = 1$. The proof of this theorem uses skew shift technique. Important: a $T$-function

$$(\chi_0, \chi_1, \chi_2, \ldots) \stackrel{F}{\mapsto} (\psi_0(\chi_0); \psi_1(\chi_0, \chi_1); \psi_2(\chi_0, \chi_1, \chi_2); \ldots),$$

is just a composition of skew shifts:

$$\begin{array}{rcl}
\chi_0 & \mapsto & \psi_0(\chi_0) \\
(\chi_0, \chi_1) & \mapsto & (\psi_0(\chi_0), \psi_1(\chi_0, \chi_1)) \\
((\chi_0, \chi_1), \chi_2) & \mapsto & ((\psi_0(\chi_0), \psi_1(\chi_0, \chi_1)), \psi_2(\chi_0, \chi_1, \chi_2)) \\
\ldots\ldots\ldots\ldots & \ldots & \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots
\end{array}$$

# Using Boolean representations

As it was said, direct use of these folklore results to verify whether a composition of arithmetic operations and bitwise logical operations is invertible (or whether it has single cycle property), is possible, but mainly for rather simple compositions.

*Note*: The bit-slice techniques of Klimov and Shamir, which they introduced in 2002, are just re-statements of the above mentioned folklore theorems.

# Using Boolean representations

For instance, with the use of these folklore theorems the following results (among others) were obtained:

- (Kotomina, 1999) The mapping

$$f(x) = (\ldots((((x + c_0) \oplus d_0) + c_1) \oplus d_1) + \cdots$$

  has a single cycle property on $n$ bit words $(n \geq 2)$ iff it has this property on 2-bit words;

- (Anashin, 2004) For any $T$-function $f$ with a single cycle property and any $T$-function $v$ the following functions have single cycle property: $f(x + 4 \cdot v(x))$, $f((x) \oplus (4 \cdot v(x)))$, $f(x) + 4 \cdot v(x)$, and $f(x) \oplus (4 \cdot v(x))$.

# Using Boolean representations

The other use of these folklore results are constructions of multivariate $T$-functions with a single cycle property.

# Multivariate $T$-functions

In 2004 Klimov and Shamir introduced a multivariate $T$-function $H$ with a single cycle property. The $m$-variate mapping

$$H : (\overrightarrow{x}_0, \overrightarrow{x}_1, \ldots, \overrightarrow{x}_{m-1}) \mapsto (h_0, h_1, \ldots, h_{m-1})$$

over $n$-bit words $\overrightarrow{x}_0, \overrightarrow{x}_1, \ldots, \overrightarrow{x}_{m-1}$, defined by

$$h_s = \overrightarrow{x}_s \oplus ((h(\overrightarrow{x}_0 \wedge \cdots \wedge \overrightarrow{x}_{m-1}) \oplus (\overrightarrow{x}_0 \wedge \cdots \wedge \overrightarrow{x}_{m-1})) \wedge \overrightarrow{x}_0 \wedge \cdots \wedge \overrightarrow{x}_{s-1},$$

$s = 0, 1, \ldots, m-1$, has a single cycle property whenever $h$ is a univariate $T$-function with a single cycle property.

# Multivariate $T$-functions

In 2004 Klimov and Shamir introduced a multivariate $T$-function $H$ with a single cycle property. In fact, this is just a trick: The $m$-variate mapping $H$ on $n$-bit words is a multivariate representation of a univariate $T$-function over $mn$-bit words.

# Multivariate $T$-functions: A trick

Given a univariate $T$-function $F$,

$$x = (\chi_0, \chi_1, \chi_2, \ldots) \overset{F}{\mapsto} (\psi_0(\chi_0); \psi_1(\chi_0, \chi_1); \psi_2(\chi_0, \chi_1, \chi_2); \ldots),$$

arrange this mapping in columns of height $m$, this way:

$$\begin{array}{llll}
\chi_0 & \chi_m & \chi_{2m} \ldots & \overset{f_0}{\mapsto} \quad \psi_0(x) \quad \psi_m(x) \quad \psi_{2m}(x) \ldots \\
\chi_1 & \chi_{m+1} & \chi_{2m+1} \ldots & \overset{f_1}{\mapsto} \quad \psi_1(x) \quad \psi_{m+1}(x) \quad \psi_{2m+1}(x) \ldots \\
\ldots & \ldots & \ldots & \ldots \\
\chi_{m-1} & \chi_{2m-1} & \chi_{3m-1} \ldots & \overset{f_{m-1}}{\mapsto} \quad \psi_{m-1}(x) \quad \psi_{2m-1}(x) \quad \psi_{3m-1}(x) \ldots
\end{array}$$

Now just assume the left-hand rows are new variables:

$$\overrightarrow{x}_j = (\chi_j, \chi_{m+j}, \chi_{2m+j}, \ldots), \qquad (j = 0, 1, \ldots, m-1)$$

# Multivariate $T$-functions: A trick

Consider the simplest example: $F(x) = 1 + x$. We have

$$\delta_j(F(x)) \equiv \delta_j(x) + \prod_{s=0}^{j-1} \delta_s(x) \pmod 2$$

(we assume the product over the empty set is 1); then the $m$-variate representation $\mathbf{F} = (f_0, f_1, \ldots, f_{m-1})$ of this mapping is

$$f_k(\overrightarrow{x}_0, \ldots, \overrightarrow{x}_{m-1}) = \overrightarrow{x}_k \oplus \left( \left( \bigwedge_{s=0}^{k-1} \overrightarrow{x}_s \right) \wedge \left( \bigwedge_{r=0}^{m-1} ((\overrightarrow{x}_r + 1) \oplus \overrightarrow{x}_r) \right) \right) =$$

$$\overrightarrow{x}_k \oplus \left( \left( \bigwedge_{s=0}^{k-1} \overrightarrow{x}_s \right) \wedge \left( \left( \left( \bigwedge_{r=0}^{m-1} \overrightarrow{x}_r \right) + 1 \right) \oplus \left( \bigwedge_{r=0}^{m-1} \overrightarrow{x}_r \right) \right) \right)$$

# Using a trick

**Proposition 2.** (Anashin, 2004) *Let* $t, j \in \{0, 1, \ldots, m-1\}$, *let all* $f_j^{(t)}$ *(resp.,* $g_j^{(t)}$*) be univariate transitive (resp, bijective) modulo* $2^n$ $T$*-functions. Then the mapping* $\mathbf{F}(\mathbf{x}) = (f_0(\mathbf{x}), \ldots, f_{m-1}(\mathbf{x}))$

$$f_0(\mathbf{x}) = \overrightarrow{x}_0 \boxplus \left( \bigwedge_{r=0}^{m-1} (f_0^{(r)}(\overrightarrow{x}_r) \oplus \overrightarrow{x}_r) \right);$$

$$f_1(\mathbf{x}) = \overrightarrow{x}_1 \boxplus \left( g_1^{(0)}(\overrightarrow{x}_0) \wedge \left( \bigwedge_{r=0}^{m-1} (f_1^{(r)}(\overrightarrow{x}_r) \oplus \overrightarrow{x}_r) \right) \right);$$

. . . . . .

$$f_{m-1}(\mathbf{x}) = \overrightarrow{x}_{m-1} \boxplus \left( \left( \bigwedge_{t=0}^{m-2} g_{m-1}^{(t)}(\overrightarrow{x}_t) \right) \wedge \left( \bigwedge_{r=0}^{m-1} (f_{m-1}^{(r)}(\overrightarrow{x}_r) \oplus \overrightarrow{x}_r) \right) \right),$$

*where* $\mathbf{x} = (\overrightarrow{x}_0, \ldots, \overrightarrow{x}_{m-1})$, $\boxplus \in \{+, \oplus\}$, *has a single cycle pr-ty*

# Coming back to $p$-adic analysis

Unfortunately, no $T$-functions with a single cycle property, which are REALLY multivariate, are known today. Among 'natural' functions these ones do not exist!

# Coming back to $p$-adic analysis

**Theorem 8.** (Anashin, 1993) *Let the function* $F = (f_1, \ldots, f_n) \colon \mathbb{Z}_p^n \to \mathbb{Z}_p^n$ *be compatible, ergodic, and uniformly differentiable modulo* $p$ *on* $\mathbb{Z}_p$. *Then* $n = 1$.

# Coming back to $p$-adic analysis

Note: Compare to differentiability, the differentiability modulo $p^k$ is a weaker restriction. In fact,

$$\frac{F(\mathbf{u}+\mathbf{h})-F(\mathbf{u})}{\mathbf{h}} \approx F'_k(\mathbf{u})$$

$\approx$ with arbitrarily high precision $\Rightarrow$ differentiability

$\approx$ with precision not worse than $p^{-k} \Rightarrow$ differentiability $\bmod p^k$

# Coming back to $p$-adic analysis

In fact we have already used uniform differentiability modulo $p^k$ when proving that some property holds modulo *all* $p^n$ whenever it holds modulo *some* $p^{n_0}$.

# Coming back to $p$-adic analysis

*Note.* (Anashin, 1993) All univariate invertible $T$-functions on $n$-bit words are just reductions modulo $2^n$ of some compatible functions on $\mathbb{Z}_2$, which are uniformly differentiable modulo $2$.

# Coming back to $p$-adic analysis

*Note.* (Anashin, 2004) Any transitive $m$-variate mapping $U : (\mathbb{Z}/2^n)^m \to (\mathbb{Z}/2^n)^m$ could be constructively (with the use of skew shifts) raised to continuous mapping $\tilde{U} : (\mathbb{Z}_2)^m \to (\mathbb{Z}_2)^m$, which is transitive modulo $2^N$ for all $N \geq n$.

# A ‘provable’ security

To prove a cipher is secure one makes a ‘polynomial-time’ reduction to one of plausible (but still unproven) conjectures of ‘intractability’ of a certain problem, which is ‘hard in average’.

Within the class of our PRNG’s thus reduction (hence, a ‘conditional proof’ of their security) is also possible

# A 'provable' security

First, we need a problem, which is plausibly hard in average.
Consider a polynomial $\psi(\chi_0, \chi_1, \ldots, \chi_{n-1})$ over $\mathbb{Z}/2$ in variables $\chi_0, \chi_1, \ldots, \chi_{n-1}$; for $m \in \mathbb{N}$ replace $\chi_j^m$ with $\chi_j$. Thus one obtains a *Boolean polynomial*, that is, an algebraic normal form, ANF, of a Boolean function. To determine whether $k$ Boolean polynomials in $n$ variables have a common zero is an $\mathcal{NP}$-complete problem.

# A ‘provable’ security

We conjecture: For $k \leq n$ it is intractable to find a solution of a system of $k$ random Boolean equations in $n$ indeterminates (under the assumption that the number of monomials in each equation is polynomially restricted).

# A ‘provable’ security

Now, given Boolean polynomials $\psi_i$, we construct a $T$-function $f$ with a single cycle property in the following way:

For $x \in \mathbb{Z}_2$ let
$\Psi_i(x) = \psi_i(\delta_0(x), \ldots, \delta_{n-1}(x)) \in \{0,1\} \subset \mathbb{Z}_2$; put

$$f(x) = (1+x) \oplus 2^{n+1} \cdot \Psi_0(x) \oplus 2^{n+2} \cdot \Psi_1(x) \oplus \cdots \oplus 2^{n+k} \cdot \Psi_{k-1}(x)$$

In view of the above mentioned folklore result (see theorems 6 and 7) this function $f$ is a $T$-function with a single cycle property.

# A ‘provable’ security

Then we construct a PRNG. Take $f \bmod 2^{n+k+1}$ as a state update function, $G = \lfloor \frac{z}{2^{n+1}} \rfloor \bmod 2^k$ (a truncation of $n+1$ low order bits) as an output function, and $x_0 \in \{0, 1, \ldots, 2^n - 1\}$ as a key.

The produced output sequence attains all the above mentioned properties (period of length $2^{n+k+1}$, uniform distribution, etc.)

# A ‘provable’ security

However, it is not difficult to show that to find a state $x = \chi_0 + \chi_1 \cdot 2 + \cdots \chi_{n-1} \cdot 2^{n-1}$ given an output, an adversary (with probability $1 - \frac{1}{2^n}$) has to solve a Boolean system

$$\psi_i(\chi_0, \chi_1, \ldots, \chi_{n-1}) = \varepsilon_i \quad (i = 1, 2, \ldots, k),$$

where $\varepsilon_i \in \{0, 1\}$ are determined by the output.

# A ‘provable’ security

Moreover, with the use of the above technique it is clear how to construct in a similar way a counter-dependent PRNG, which produces an output sequence that attains all the above mentioned properties.

That is, at each new step an adversary will have to solve a new system of Boolean equations, i.e., the left hand part of a system will change from step to step.

# Conclusions

On the one hand, it is possible to build fast and secure stream ciphers based on 2-adic ergodic functions: Our schemes attain performance 6 Gbit per second at 3 GHz Intel P4 processor.

Use of these functions results in new cryptographic properties, which make the cipher more secure: First of all, this is a possibility of making the functions key-dependent, and changing them dynamically during the encryption.

# Conclusions

On the other hand, one must be very careful when choosing $T$-functions for a stream cipher: Too many of these functions are fast, yet bad. One bad function among others good in a composition of a (counter-dependent) PRNG is enough to spoil the whole cipher!

# Conclusions

Cryptographic properties of $T$-functions are tightly connected with specific features of the corresponding non-Archimedean dynamics.

These dynamics are rich, intriguing, and worth deeper study to develop new fast and secure ciphers.