Stream ciphers

Stream ciphers

Stream Ciphers are Psuedorandom Generators made practical!

Stream ciphers

Stream Ciphers are Psuedorandom Generators made practical!

They are better than PRG's!

Stream ciphers

Stream Ciphers are Psuedorandom Generators made practical!

They are better than PRG's!

Are Stream Ciphers ciphers? Depends on who you ask.

Stream ciphers

Stream Ciphers are Psuedorandom Generators made practical!

They are better than PRG's!

Are Stream Ciphers ciphers? Depends on who you ask.

Some people identify the stream cipher with the cipher that results from using it as the pseudo-one-time-pad.

Stream ciphers

Stream Ciphers are Psuedorandom Generators made practical!

They are better than PRG's!

Are Stream Ciphers ciphers? Depends on who you ask.

Some people identify the stream cipher with the cipher that results from using it as the pseudo-one-time-pad.

We will not do that.

Stream ciphers

Stream Ciphers are Psuedorandom Generators made practical!

They are better than PRG's!

Are Stream Ciphers ciphers? Depends on who you ask.

Some people identify the stream cipher with the cipher that results from using it as the pseudo-one-time-pad.

We will not do that.

However,

Stream ciphers

Stream Ciphers are Psuedorandom Generators made practical!

They are better than PRG's!

Are Stream Ciphers ciphers? Depends on who you ask.

Some people identify the stream cipher with the cipher that results from using it as the pseudo-one-time-pad.

We will not do that.

However,

we are right, and they are wrong.

Stream ciphers

- As we defined them, PRGs are limited
  - They have fixed-length output
  - They produce output in "one shot"

Stream ciphers

- As we defined them, PRGs are limited
  - They have fixed-length output
  - They produce output in "one shot"
- In practice, Psuedo 1-Time Pads use **Stream Ciphers**
  - Can be viewed as producing an "infinite" stream of pseudorandom bits, on demand
  - More flexible, more efficient

Stream ciphers

A **Stream Cipher** is basically a **recurrence** that generates bits.
Formally a **Stream Cipher** is a pair of efficient, deterministic algorithms (Init, GetBits) such that:

1. Init does the following:
   1.1 **Input** **private** seed s. Think of as truly random.
   1.2 **Output** $y_0, y_1, \ldots, y_n$ for some n.

Stream ciphers

A **Stream Cipher** is basically a **recurrence** that generates bits.
Formally a **Stream Cipher** is a pair of efficient, deterministic algorithms (Init, GetBits) such that:

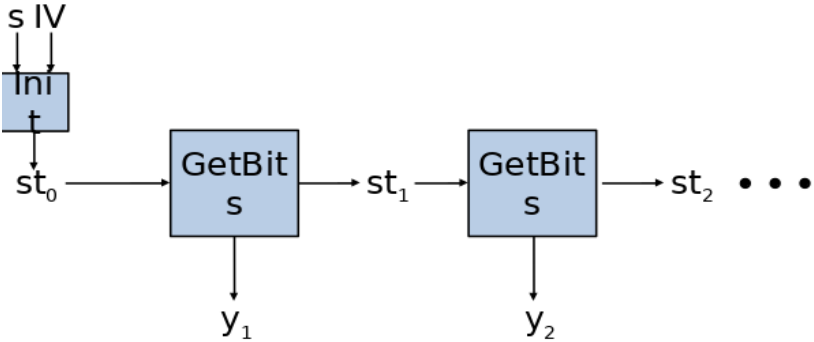
1. Init does the following:
   1.1 **Input** **private** seed s. Think of as truly random.
   1.2 **Output** $y_0, y_1, \ldots, y_n$ for some n.

2. GetBits does the following:
   2.1 **Input** Given $y_0, \ldots, y_m$ (likely depends on less of the past).
   2.2 **Output** the bit y_{m+1}.

Note In practice, y_i is a block rather than a bit.

Stream ciphers

- Can use (Init, GetBits) to generate any desired number of output bits from an initial seed



Stream ciphers

- A stream cipher is **secure** (informally) if the output stream generated from a uniform seed is pseudorandom
  - I.e. regardless of how long the output stream is (so long as it is polynomial)
  - We omit formal definition which is in terms of games.

Do Stream Ciphers exist? Theoretical

Under reasonable crypto assumptions can construct Secure Stream Cipher.

Do Stream Ciphers exist? Theoretical

Under reasonable crypto assumptions can construct Secure Stream Cipher.

A stream cipher constructed this way is too slow to really use.

Do Stream Ciphers exist? Theoretical

Under reasonable crypto assumptions can construct Secure Stream Cipher.

A stream cipher constructed this way is too slow to really use.

Still good to have proof-of-concept.

Do Stream Ciphers exist? Theoretical

Under reasonable crypto assumptions can construct Secure Stream Cipher.

A stream cipher constructed this way is too slow to really use.

Still good to have proof-of-concept.

Over time, constructions that are **too slow** are worked on and become fast enough.

Do Stream Ciphers exist? Practical

Attempts at Stream Ciphers:

Do Stream Ciphers exist? Practical

Attempts at Stream Ciphers:

1. Linear Feedback Shift Registers. Fast! Used! Not Secure!

Do Stream Ciphers exist? Practical

Attempts at Stream Ciphers:

1. Linear Feedback Shift Registers. Fast! Used! Not Secure!
2. Trivium. Fast! Used! Empirically Secure! Not proven.

Do Stream Ciphers exist? Practical

Attempts at Stream Ciphers:

1. Linear Feedback Shift Registers. Fast! Used! Not Secure!
2. Trivium. Fast! Used! Empirically Secure! Not proven.
3. Rivest Cipher 4. Fast! Used! No longer secure!

Do Stream Ciphers exist? Practical

Attempts at Stream Ciphers:

1. Linear Feedback Shift Registers. Fast! Used! Not Secure!
2. Trivium. Fast! Used! Empirically Secure! Not proven.
3. Rivest Cipher 4. Fast! Used! No longer secure!

Note Seems impossible to get Stream Ciphers that are provably (even using Hardness Assumptions) secure and practical.

Do Stream Ciphers exist? Practical

Attempts at Stream Ciphers:

1. Linear Feedback Shift Registers. Fast! Used! Not Secure!
2. Trivium. Fast! Used! Empirically Secure! Not proven.
3. Rivest Cipher 4. Fast! Used! No longer secure!

Note Seems impossible to get Stream Ciphers that are provably (even using Hardness Assumptions) secure and practical.

Note But having the rigor gives the practitioners (1) a target to shoot for, and (2) pitfalls to watch out for.

Linear Feedback Shift Registers (LFSR): Example

Degree 3 LFSR, 3 constants : $c_3, c_2, c_1 \in \{0, 1\}$. $+$ is mod 2.

Key is 3 bits: (y_0, y_1, y_2).

Linear Feedback Shift Registers (LFSR): Example

Degree 3 LFSR, 3 constants : $c_3, c_2, c_1 \in \{0, 1\}$. $+$ is mod 2.

Key is 3 bits: (y_0, y_1, y_2).

$$(\forall t \geq 3)[y_t = c_1 y_{t-1} + c_2 y_{t-2} + c_3 y_{t-3}].$$

Note Leave it to you to generalize to degree n LFSR.

LFSRs

LFSRs

1. Will eventually be periodic but hope the periodicity is long.

LFSRs

1. Will eventually be periodic but hope the periodicity is long.
2. For n-degree max periodicity is $2^n - 1$.

LFSRs

1. Will eventually be periodic but hope the periodicity is long.
2. For n-degree max periodicity is $2^n - 1$.
3. Known how to set feedback coefficients so as to achieve $2^n - 1$.

LFSRs

1. Will eventually be periodic but hope the periodicity is long.
2. For n-degree max periodicity is $2^n - 1$.
3. Known how to set feedback coefficients so as to achieve $2^n - 1$.
4. Maximal-length LFSRs have good statistical properties.

LFSRs

1. Will eventually be periodic but hope the periodicity is long.
2. For n-degree max periodicity is $2^n - 1$.
3. Known how to set feedback coefficients so as to achieve $2^n - 1$.
4. Maximal-length LFSRs have good statistical properties.
5. Are LFSRs secure? Vote YES, NO, UNKNOWN TO SCIENCE.

LFSRs

1. Will eventually be periodic but hope the periodicity is long.
2. For n-degree max periodicity is $2^n - 1$.
3. Known how to set feedback coefficients so as to achieve $2^n - 1$.
4. Maximal-length LFSRs have good statistical properties.
5. Are LFSRs secure? Vote YES, NO, UNKNOWN TO SCIENCE. NO.

Example of Bad Security

Degree 3. c_0, c_1, c_2 unknown. If $y_1, y_2, y_3, y_4, y_5, y_6$ become known then:

Example of Bad Security

Degree 3. c_0, c_1, c_2 unknown. If $y_1, y_2, y_3, y_4, y_5, y_6$ become known then:

$y_4 = c_2 y_3 + c_1 y_2 + c_0 y_1$
$y_5 = c_2 y_4 + c_1 y_3 + c_0 y_2$
$y_6 = c_2 y_5 + c_1 y_4 + c_0 y_3$

Example of Bad Security

Degree 3. c_0, c_1, c_2 unknown. If $y_1, y_2, y_3, y_4, y_5, y_6$ become known then:

$y_4 = c_2 y_3 + c_1 y_2 + c_0 y_1$
$y_5 = c_2 y_4 + c_1 y_3 + c_0 y_2$
$y_6 = c_2 y_5 + c_1 y_4 + c_0 y_3$

3 linear equations in 3 variables. Can find c_0, c_1, c_2. **Cracked!**

For n-degree LFSR can crack after $2n$ iterations.
Moral: Linearity is *bad* cryptography.

LFSR and Linearity

Linearity makes LFSR's **fast**

LFSR and Linearity

Linearity makes LFSR's **fast**

Linearity makes LFSR's **crackable**

LFSR and Linearity

Linearity makes LFSR's **fast**

Linearity makes LFSR's **crackable**

Who first said:

Those who lives by linearity, dies by linearity!

LFSR and Linearity

Linearity makes LFSR's **fast**

Linearity makes LFSR's **crackable**

Who first said:

> **Those who lives by linearity, dies by linearity!**

It was Irene!

The Essence of Crypto

Recall: The Essence of Crypto is to make computation

The Essence of Crypto

Recall: The Essence of Crypto is to make computation

1. Easy for Alice and Bob.

The Essence of Crypto

Recall: The Essence of Crypto is to make computation

1. Easy for Alice and Bob.
2. Hard for Eve.

The Essence of Crypto

Recall: The Essence of Crypto is to make computation

1. Easy for Alice and Bob.
2. Hard for Eve.

LFSR makes computation easy for all three!

Nonlinear Feedback Shift Registers (NFSRs)

Nonlinear Feedback Shift Registers (NFSRs)

1. Add nonlinearity to prevent attacks

Nonlinear Feedback Shift Registers (NFSRs)

1. Add nonlinearity to prevent attacks
2. Nonlinear feedback

Nonlinear Feedback Shift Registers (NFSRs)

1. Add nonlinearity to prevent attacks
2. Nonlinear feedback
3. Output is a nonlinear function of the state

Nonlinear Feedback Shift Registers (NFSRs)

1. Add nonlinearity to prevent attacks
2. Nonlinear feedback
3. Output is a nonlinear function of the state
4. Multiple (coupled) LFSRs

Nonlinear Feedback Shift Registers (NFSRs)

1. Add nonlinearity to prevent attacks
2. Nonlinear feedback
3. Output is a nonlinear function of the state
4. Multiple (coupled) LFSRs
5. . . . or any combination of the above

Nonlinear Feedback Shift Registers (NFSRs)

1. Add nonlinearity to prevent attacks
2. Nonlinear feedback
3. Output is a nonlinear function of the state
4. Multiple (coupled) LFSRs
5. . . . or any combination of the above
6. Still want to preserve statistical properties of the output, and long cycle length

Nonlinear Feedback Shift Registers

Assume n even. $+$ is mod 2.
Initialize with x_1, x_2, x_3, x_4
$(\forall n \geq 5)[x_n = x_{n-1}x_{n-2} + x_{n-2}x_{n-3} + x_{n-3}x_{n-4}]$.

Nonlinear Feedback Shift Registers

Assume n even. $+$ is mod 2.
Initialize with x_1, x_2, x_3, x_4
$(\forall n \geq 5)[x_n = x_{n-1}x_{n-2} + x_{n-2}x_{n-3} + x_{n-3}x_{n-4}]$.

Is this a good stream cipher? **Vote** Y (with HA), N, UN

Nonlinear Feedback Shift Registers

Assume n even. $+$ is mod 2.
Initialize with x_1, x_2, x_3, x_4
$(\forall n \geq 5)[x_n = x_{n-1}x_{n-2} + x_{n-2}x_{n-3} + x_{n-3}x_{n-4}]$.

Is this a good stream cipher? **Vote** Y (with HA), N, UN

UNKNOWN

Nonlinear Feedback Shift Registers

Assume n even. $+$ is mod 2.
Initialize with x_1, x_2, x_3, x_4
$(\forall n \geq 5)[x_n = x_{n-1}x_{n-2} + x_{n-2}x_{n-3} + x_{n-3}x_{n-4}]$.

Is this a good stream cipher? **Vote** Y (with HA), N, UN

UNKNOWN
I made up this cipher last year for example of nonlinear.
On the HW you will tell me if its a good stream cipher.

Trivum

Trivium

- Designed by De Cannière and Preneel in 2006 as part of eSTREAM competition.

Trivium

- Designed by De Cannière and Preneel in 2006 as part of eSTREAM competition.

- Intended to be simple and efficient (especially in hardware).

Trivium

- Designed by De Cannière and Preneel in 2006 as part of eSTREAM competition.
- Intended to be simple and efficient (especially in hardware).
- Essentially no attacks better than brute-force search are known.

Trivium

- Three coupled Feedback Shift Registers (FSR) of degree 93, 84, and 111.
- Initialization:
  - 80-bit key in left-most registers of first FSR. This is private.

Trivium

- Three coupled Feedback Shift Registers (FSR) of degree 93, 84, and 111.
- Initialization:
  - 80-bit key in left-most registers of first FSR. This is private.
  - 80-bit IV in left-most registers of second FSR. This is public.

Trivium

- Three coupled Feedback Shift Registers (FSR) of degree 93, 84, and 111.
- Initialization:
  - 80-bit key in left-most registers of first FSR. This is private.
  - 80-bit IV in left-most registers of second FSR. This is public.
  - Remaining registers set to 0, except for three right-most registers of third FSR

Trivium

- Three coupled Feedback Shift Registers (FSR) of degree 93, 84, and 111.
- Initialization:
  - 80-bit key in left-most registers of first FSR. This is private.
  - 80-bit IV in left-most registers of second FSR. This is public.
  - Remaining registers set to 0, except for three right-most registers of third FSR
  - Run for 4 x 288 clock ticks to finish init.

Trivium-Initialization

Trivium-Initialization

$K_1, \ldots, K_{80}$ Random

Trivium-Initialization

$K_1, \ldots, K_{80}$ Random
$IV_1, \ldots, IV_{80}$ Random

Trivium-Initialization

$K_1, \ldots, K_{80}$ Random
$IV_1, \ldots, IV_{80}$ Random
$(a_1, \ldots, a_{93}) \leftarrow (K_1, \ldots, K_{80}, 0, \ldots, 0)$

Trivium-Initialization

$K_1, \ldots, K_{80}$ Random
$IV_1, \ldots, IV_{80}$ Random
$(a_1, \ldots, a_{93}) \leftarrow (K_1, \ldots, K_{80}, 0, \ldots, 0)$
$(b_1, \ldots, b_{84}) \leftarrow (IV_1, \ldots, IV_{80}, 0, 0, 0, 0)$

Trivium-Initialization

$K_1, \ldots, K_{80}$ Random
$IV_1, \ldots, IV_{80}$ Random
$(a_1, \ldots, a_{93}) \leftarrow (K_1, \ldots, K_{80}, 0, \ldots, 0)$
$(b_1, \ldots, b_{84}) \leftarrow (IV_1, \ldots, IV_{80}, 0, 0, 0, 0)$
$(c_1, \ldots, c_{111}) \leftarrow (0, \ldots, 0, 1, 1, 1)$

Trivium-Initialization

$K_1, \ldots, K_{80}$ Random
$IV_1, \ldots, IV_{80}$ Random
$(a_1, \ldots, a_{93}) \leftarrow (K_1, \ldots, K_{80}, 0, \ldots, 0)$
$(b_1, \ldots, b_{84}) \leftarrow (IV_1, \ldots, IV_{80}, 0, 0, 0, 0)$
$(c_1, \ldots, c_{111}) \leftarrow (0, \ldots, 0, 1, 1, 1)$
For $i = 1$ to 4×288 do

Trivium-Initialization

$K_1, \ldots, K_{80}$ Random
$IV_1, \ldots, IV_{80}$ Random
$(a_1, \ldots, a_{93}) \leftarrow (K_1, \ldots, K_{80}, 0, \ldots, 0)$
$(b_1, \ldots, b_{84}) \leftarrow (IV_1, \ldots, IV_{80}, 0, 0, 0, 0)$
$(c_1, \ldots, c_{111}) \leftarrow (0, \ldots, 0, 1, 1, 1)$
For $i = 1$ to 4×288 do

1. $t_1 \leftarrow a_{86} + a_{91} a_{92} + b_{79}$

Trivium-Initialization

$K_1, \ldots, K_{80}$ Random
$IV_1, \ldots, IV_{80}$ Random
$(a_1, \ldots, a_{93}) \leftarrow (K_1, \ldots, K_{80}, 0, \ldots, 0)$
$(b_1, \ldots, b_{84}) \leftarrow (IV_1, \ldots, IV_{80}, 0, 0, 0, 0)$
$(c_1, \ldots, c_{111}) \leftarrow (0, \ldots, 0, 1, 1, 1)$
For $i = 1$ to 4×288 do

1. $t_1 \leftarrow a_{86} + a_{91} a_{92} + b_{79}$
2. $t_2 \leftarrow b_{70} + b_{83} b_{84} + c_1 + c_{87}$

Trivium-Initialization

$K_1, \ldots, K_{80}$ Random
$IV_1, \ldots, IV_{80}$ Random
$(a_1, \ldots, a_{93}) \leftarrow (K_1, \ldots, K_{80}, 0, \ldots, 0)$
$(b_1, \ldots, b_{84}) \leftarrow (IV_1, \ldots, IV_{80}, 0, 0, 0, 0)$
$(c_1, \ldots, c_{111}) \leftarrow (0, \ldots, 0, 1, 1, 1)$
For $i = 1$ to 4×288 do

1. $t_1 \leftarrow a_{86} + a_{91}a_{92} + b_{79}$
2. $t_2 \leftarrow b_{70} + b_{83}b_{84} + c_1 + c_{87}$
3. $t_3 \leftarrow c_{66} + c_{100}c_{110} + c_{111} + a_{69}$

Trivium-Initialization

$K_1, \ldots, K_{80}$ Random
$IV_1, \ldots, IV_{80}$ Random
$(a_1, \ldots, a_{93}) \leftarrow (K_1, \ldots, K_{80}, 0, \ldots, 0)$
$(b_1, \ldots, b_{84}) \leftarrow (IV_1, \ldots, IV_{80}, 0, 0, 0, 0)$
$(c_1, \ldots, c_{111}) \leftarrow (0, \ldots, 0, 1, 1, 1)$
For $i = 1$ to 4×288 do

1. $t_1 \leftarrow a_{86} + a_{91} a_{92} + b_{79}$
2. $t_2 \leftarrow b_{70} + b_{83} b_{84} + c_1 + c_{87}$
3. $t_3 \leftarrow c_{66} + c_{100} c_{110} + c_{111} + a_{69}$
4. $(a_1, \ldots, a_{93}) \leftarrow (t_3, a_1, \ldots, a_{92})$

Trivium-Initialization

$K_1, \ldots, K_{80}$ Random
$IV_1, \ldots, IV_{80}$ Random
$(a_1, \ldots, a_{93}) \leftarrow (K_1, \ldots, K_{80}, 0, \ldots, 0)$
$(b_1, \ldots, b_{84}) \leftarrow (IV_1, \ldots, IV_{80}, 0, 0, 0, 0)$
$(c_1, \ldots, c_{111}) \leftarrow (0, \ldots, 0, 1, 1, 1)$
For $i = 1$ to 4×288 do

1. $t_1 \leftarrow a_{86} + a_{91} a_{92} + b_{79}$
2. $t_2 \leftarrow b_{70} + b_{83} b_{84} + c_1 + c_{87}$
3. $t_3 \leftarrow c_{66} + c_{100} c_{110} + c_{111} + a_{69}$
4. $(a_1, \ldots, a_{93}) \leftarrow (t_3, a_1, \ldots, a_{92})$
5. $(b_1, \ldots, b_{83}) \leftarrow (t_1, b_1, \ldots, b_{82})$

Trivium-Initialization

$K_1, \ldots, K_{80}$ Random
$IV_1, \ldots, IV_{80}$ Random
$(a_1, \ldots, a_{93}) \leftarrow (K_1, \ldots, K_{80}, 0, \ldots, 0)$
$(b_1, \ldots, b_{84}) \leftarrow (IV_1, \ldots, IV_{80}, 0, 0, 0, 0)$
$(c_1, \ldots, c_{111}) \leftarrow (0, \ldots, 0, 1, 1, 1)$
For $i = 1$ to 4×288 do

1. $t_1 \leftarrow a_{86} + a_{91} a_{92} + b_{79}$
2. $t_2 \leftarrow b_{70} + b_{83} b_{84} + c_1 + c_{87}$
3. $t_3 \leftarrow c_{66} + c_{100} c_{110} + c_{111} + a_{69}$
4. $(a_1, \ldots, a_{93}) \leftarrow (t_3, a_1, \ldots, a_{92})$
5. $(b_1, \ldots, b_{83}) \leftarrow (t_1, b_1, \ldots, b_{82})$
6. $(c_1, \ldots, c_{111}) \leftarrow (t_2, c_1, \ldots, c_{110})$

Trivium-Initialization

$K_1, \dots, K_{80}$ Random
$IV_1, \dots, IV_{80}$ Random
$(a_1, \dots, a_{93}) \leftarrow (K_1, \dots, K_{80}, 0, \dots, 0)$
$(b_1, \dots, b_{84}) \leftarrow (IV_1, \dots, IV_{80}, 0, 0, 0, 0)$
$(c_1, \dots, c_{111}) \leftarrow (0, \dots, 0, 1, 1, 1)$
For $i = 1$ to 4×288 do

1. $t_1 \leftarrow a_{86} + a_{91}a_{92} + b_{79}$
2. $t_2 \leftarrow b_{70} + b_{83}b_{84} + c_1 + c_{87}$
3. $t_3 \leftarrow c_{66} + c_{100}c_{110} + c_{111} + a_{69}$
4. $(a_1, \dots, a_{93}) \leftarrow (t_3, a_1, \dots, a_{92})$
5. $(b_1, \dots, b_{83}) \leftarrow (t_1, b_1, \dots, b_{82})$
6. $(c_1, \dots, c_{111}) \leftarrow (t_2, c_1, \dots, c_{110})$

Note No random bits output. This is just initialization.

Trivium-Iteration

We omit superscripts for readability.

Trivium-Iteration

We omit superscripts for readability.

For $i = 1$ to ∞ do

Trivium-Iteration

We omit superscripts for readability.
For $i = 1$ to ∞ do

1. $y_i = a_{66} + a_{93} + b_{70} + b_{75} + c_{66} + c_{111}$ (ith random bit).

Trivium-Iteration

We omit superscripts for readability.
For $i = 1$ to ∞ do

1. $y_i = a_{66} + a_{93} + b_{70} + b_{75} + c_{66} + c_{111}$ (ith random bit).
2. $t_1 \leftarrow a_{86} + a_{91} a_{92} + b_{79}$

Trivium-Iteration

We omit superscripts for readability.
For $i = 1$ to ∞ do

1. $y_i = a_{66} + a_{93} + b_{70} + b_{75} + c_{66} + c_{111}$ (ith random bit).
2. $t_1 \leftarrow a_{86} + a_{91}a_{92} + b_{79}$
3. $t_2 \leftarrow b_{70} + b_{83}b_{84} + c_1 + c_{87}$

Trivium-Iteration

We omit superscripts for readability.
For $i = 1$ to ∞ do

1. $y_i = a_{66} + a_{93} + b_{70} + b_{75} + c_{66} + c_{111}$ (ith random bit).
2. $t_1 \leftarrow a_{86} + a_{91} a_{92} + b_{79}$
3. $t_2 \leftarrow b_{70} + b_{83} b_{84} + c_1 + c_{87}$
4. $t_3 \leftarrow c_{66} + c_{100} c_{110} + c_{111} + a_{69}$

Trivium-Iteration

We omit superscripts for readability.
For $i = 1$ to ∞ do

1. $y_i = a_{66} + a_{93} + b_{70} + b_{75} + c_{66} + c_{111}$ (ith random bit).
2. $t_1 \leftarrow a_{86} + a_{91}a_{92} + b_{79}$
3. $t_2 \leftarrow b_{70} + b_{83}b_{84} + c_1 + c_{87}$
4. $t_3 \leftarrow c_{66} + c_{100}c_{110} + c_{111} + a_{69}$
5. $(a_1, \ldots, a_{93}) \leftarrow (t_3, a_1, \ldots, a_{92})$

Trivium-Iteration

We omit superscripts for readability.
For $i = 1$ to ∞ do

1. $y_i = a_{66} + a_{93} + b_{70} + b_{75} + c_{66} + c_{111}$ (ith random bit).
2. $t_1 \leftarrow a_{86} + a_{91}a_{92} + b_{79}$
3. $t_2 \leftarrow b_{70} + b_{83}b_{84} + c_1 + c_{87}$
4. $t_3 \leftarrow c_{66} + c_{100}c_{110} + c_{111} + a_{69}$
5. $(a_1, \ldots, a_{93}) \leftarrow (t_3, a_1, \ldots, a_{92})$
6. $(b_1, \ldots, b_{83}) \leftarrow (t_1, b_1, \ldots, s_{83})$

Trivium-Iteration

We omit superscripts for readability.
For $i = 1$ to ∞ do

1. $y_i = a_{66} + a_{93} + b_{70} + b_{75} + c_{66} + c_{111}$ (ith random bit).
2. $t_1 \leftarrow a_{86} + a_{91}a_{92} + b_{79}$
3. $t_2 \leftarrow b_{70} + b_{83}b_{84} + c_1 + c_{87}$
4. $t_3 \leftarrow c_{66} + c_{100}c_{110} + c_{111} + a_{69}$
5. $(a_1, \ldots, a_{93}) \leftarrow (t_3, a_1, \ldots, a_{92})$
6. $(b_1, \ldots, b_{83}) \leftarrow (t_1, b_1, \ldots, s_{83})$
7. $(c_1, \ldots, c_{111}) \leftarrow (t_2, c_1, \ldots, c_{110})$

Trivium-Iteration

We omit superscripts for readability.
For $i = 1$ to ∞ do

1. $y_i = a_{66} + a_{93} + b_{70} + b_{75} + c_{66} + c_{111}$ (ith random bit).
2. $t_1 \leftarrow a_{86} + a_{91}a_{92} + b_{79}$
3. $t_2 \leftarrow b_{70} + b_{83}b_{84} + c_1 + c_{87}$
4. $t_3 \leftarrow c_{66} + c_{100}c_{110} + c_{111} + a_{69}$
5. $(a_1, \ldots, a_{93}) \leftarrow (t_3, a_1, \ldots, a_{92})$
6. $(b_1, \ldots, b_{83}) \leftarrow (t_1, b_1, \ldots, s_{83})$
7. $(c_1, \ldots, c_{111}) \leftarrow (t_2, c_1, \ldots, c_{110})$

Note the three diff parts of s are three coupled nonlinear FSR.

Trivium based on LFSR though not LFSR

Trivium based on LFSR though not LFSR

Note:

Trivium based on LFSR though not LFSR

Note:

1. t_1, t_2, t_3 are nonlinear combos of prior bits.

Trivium based on LFSR though not LFSR

Note:

1. t_1, t_2, t_3 are nonlinear combos of prior bits.
2. $(a_1, \ldots, a_{93}) \leftarrow (t_3, a_1, \ldots, a_{92})$

Trivium based on LFSR though not LFSR

Note:

1. t_1, t_2, t_3 are nonlinear combos of prior bits.
2. $(a_1, \ldots, a_{93}) \leftarrow (t_3, a_1, \ldots, a_{92})$
3. $(b_1, \ldots, b_{83}) \leftarrow (t_1, s_1 \ldots, s_{82})$

Trivium based on LFSR though not LFSR

Note:

1. t_1, t_2, t_3 are nonlinear combos of prior bits.
2. $(a_1, \ldots, a_{93}) \leftarrow (t_3, a_1, \ldots, a_{92})$
3. $(b_1, \ldots, b_{83}) \leftarrow (t_1, s_1 \ldots, s_{82})$
4. $(c_1, \ldots, c_{111}) \leftarrow (t_2, c_1, \ldots, c_{110})$

Trivium based on LFSR though not LFSR

Note:

1. t_1, t_2, t_3 are nonlinear combos of prior bits.
2. $(a_1, \ldots, a_{93}) \leftarrow (t_3, a_1, \ldots, a_{92})$
3. $(b_1, \ldots, b_{83}) \leftarrow (t_1, s_1 \ldots, s_{82})$
4. $(c_1, \ldots, c_{111}) \leftarrow (t_2, c_1, \ldots, c_{110})$

Since t_1, t_2, t_3 nonlinear, Trivium is NOT LFSR

Trivium based on LFSR though not LFSR

Note:

1. t_1, t_2, t_3 are nonlinear combos of prior bits.
2. $(a_1, \ldots, a_{93}) \leftarrow (t_3, a_1, \ldots, a_{92})$
3. $(b_1, \ldots, b_{83}) \leftarrow (t_1, s_1 \ldots, s_{82})$
4. $(c_1, \ldots, c_{111}) \leftarrow (t_2, c_1, \ldots, c_{110})$

Since t_1, t_2, t_3 nonlinear, Trivium is NOT LFSR
But is very much like LFSR.

Trivium based on LFSR though not LFSR

Note:

1. t_1, t_2, t_3 are nonlinear combos of prior bits.
2. $(a_1, \ldots, a_{93}) \leftarrow (t_3, a_1, \ldots, a_{92})$
3. $(b_1, \ldots, b_{83}) \leftarrow (t_1, s_1 \ldots, s_{82})$
4. $(c_1, \ldots, c_{111}) \leftarrow (t_2, c_1, \ldots, c_{110})$

Since t_1, t_2, t_3 nonlinear, Trivium is NOT LFSR
But is very much like LFSR.

Benefit: Shifting is Fast!

Facts About Trivium

Facts About Trivium

1) Has been build in hardware with 3488 logic gates. Small! Fast!

Facts About Trivium

1) Has been build in hardware with 3488 logic gates. Small! Fast!

2) So far has not been broken. That we know of!

Facts About Trivium

1) Has been build in hardware with 3488 logic gates. Small! Fast!
2) So far has not been broken. That we know of!
3) Naive method is 2^{80} steps. Guess all keys.

Facts About Trivium

1) Has been build in hardware with 3488 logic gates. Small! Fast!
2) So far has not been broken. That we know of!
3) Naive method is 2^{80} steps. Guess all keys.
4) If only do ~ 700 init steps then Cube Attack is 2^{68} steps.

Facts About Trivium

1) Has been build in hardware with 3488 logic gates. Small! Fast!
2) So far has not been broken. That we know of!
3) Naive method is 2^{80} steps. Guess all keys.
4) If only do ~ 700 init steps then Cube Attack is 2^{68} steps.
5) Seems to have long period but hard to know:

Facts About Trivium

1) Has been build in hardware with 3488 logic gates. Small! Fast!
2) So far has not been broken. That we know of!
3) Naive method is 2^{80} steps. Guess all keys.
4) If only do $\sim$ 700 init steps then Cube Attack is 2^{68} steps.
5) Seems to have long period but hard to know:
   1. Nonlin makes it hard to predict. Good for practical A and B.

Facts About Trivium

1) Has been build in hardware with 3488 logic gates. Small! Fast!

2) So far has not been broken. That we know of!

3) Naive method is 2^{80} steps. Guess all keys.

4) If only do $\sim$ 700 init steps then Cube Attack is 2^{68} steps.

5) Seems to have long period but hard to know:

   1. Nonlin makes it hard to predict. Good for practical A and B.
   2. Nonlin makes it hard to analyze. Bad for theorists A and B.

Facts About Trivium

1) Has been build in hardware with 3488 logic gates. Small! Fast!

2) So far has not been broken. That we know of!

3) Naive method is 2^{80} steps. Guess all keys.

4) If only do ~ 700 init steps then Cube Attack is 2^{68} steps.

5) Seems to have long period but hard to know:

   1. Nonlin makes it hard to predict. Good for practical A and B.
   2. Nonlin makes it hard to analyze. Bad for theorists A and B.

6) Trivium is also the name of a rock band!

Facts About Trivium

1) Has been build in hardware with 3488 logic gates. Small! Fast!
2) So far has not been broken. That we know of!
3) Naive method is 2^{80} steps. Guess all keys.
4) If only do ~ 700 init steps then Cube Attack is 2^{68} steps.
5) Seems to have long period but hard to know:
   1. Nonlin makes it hard to predict. Good for practical A and B.
   2. Nonlin makes it hard to analyze. Bad for theorists A and B.
6) Trivium is also the name of a rock band!
7) Two Papers on Trivium on course website

Why the Name Trivium?

We quote the paper

Why the Name Trivium?

We quote the paper

> *The word* **trivium** *is Latin for* **the three-fold way** *, and refers to the three-fold symmetry of TRIVIUM. The adjective* **trivial** *which was derived from it, has a connotation of simplicity, which is also one of the characteristics of TRIVIUM.*

Why the Name Trivium?

We quote the paper

> *The word* **trivium** *is Latin for* **the three-fold way** *, and refers to the three-fold symmetry of TRIVIUM. The adjective* **trivial** *which was derived from it, has a connotation of simplicity, which is also one of the characteristics of TRIVIUM.*

My Blog Post Asking if Trivium is used

This Fall I am teaching the senior course in Crypto at UMCP. Its a nice change of pace for me since REAL people REALLY use this stuff!

My Blog Post Asking if Trivium is used

This Fall I am teaching the senior course in Crypto at UMCP. Its a nice change of pace for me since REAL people REALLY use this stuff!

There is one topic that **looks** *really practical but I could not find on the web if it is or not. A Secure Stream Cipher is (informally) a way to, given a seed and optionally an Init Vector (IV), generate bits that look random.* **Trivium** *seems to be one such. According to the Trivium wiki*

THEN I HAD STUFF ABOUT TRIVIUM

My Blog Post Asking if Trivium is used

This Fall I am teaching the senior course in Crypto at UMCP. Its a nice change of pace for me since REAL people REALLY use this stuff!

There is one topic that **looks** *really practical but I could not find on the web if it is or not. A Secure Stream Cipher is (informally) a way to, given a seed and optionally an Init Vector (IV), generate bits that look random.* **Trivium** *seems to be one such. According to the Trivium wiki*

THEN I HAD STUFF ABOUT TRIVIUM

Is Trivium used?

My Blog Post Asking if Trivium is used

This Fall I am teaching the senior course in Crypto at UMCP. Its a nice change of pace for me since REAL people REALLY use this stuff!

There is one topic that **looks** *really practical but I could not find on the web if it is or not. A Secure Stream Cipher is (informally) a way to, given a seed and optionally an Init Vector (IV), generate bits that look random.* **Trivium** *seems to be one such. According to the Trivium wiki*

THEN I HAD STUFF ABOUT TRIVIUM

Is Trivium used?
If so then by whom and for what (for the psuedo 1-time pad?) ?

My Blog Post Asking if Trivium is used

This Fall I am teaching the senior course in Crypto at UMCP. Its a nice change of pace for me since REAL people REALLY use this stuff!

There is one topic that **looks** *really practical but I could not find on the web if it is or not. A Secure Stream Cipher is (informally) a way to, given a seed and optionally an Init Vector (IV), generate bits that look random.* **Trivium** *seems to be one such. According to the Trivium wiki*

THEN I HAD STUFF ABOUT TRIVIUM

Is Trivium used?
If so then by whom and for what (for the psuedo 1-time pad?) ?
If not then why not?

First Comment on Blog

Great post on Trivial! Hardware Cube Attack. Click HERE to buy Trivial Pursuit Deluxe edition!

First Comment on Blog

Great post on Trivial! Hardware Cube Attack. Click HERE to buy Trivial Pursuit Deluxe edition!

I blocked the comment as it was clearly spam, and not very good spam at that.

First Comment on Blog

Great post on Trivial! Hardware Cube Attack. Click HERE to buy Trivial Pursuit Deluxe edition!

I blocked the comment as it was clearly spam, and not very good spam at that.

Too bad. They called my post **Great** .

Second Comment on Blog

An 80-bit key/IV is not secure enough for many modern uses (like encryption on the Internet), though I am not sure what exactly Trivium and other "lightweight ciphers" consider a threat. Their primary intended deployment scenarios are IoT and hardware tokens like auto door locks. For that purpose it is secure.

Salsa20 Stream Cipher

Notation: $\oplus$ is the usual bit-wise XOR. $+$ is mod 2^{32} addition. $<<<$ will mean you circular shift bits to the left.

Salsa20 Stream Cipher

Notation: $\oplus$ is the usual bit-wise XOR. $+$ is mod 2^{32} addition.
$<<<$ will mean you circular shift bits to the left.
Basic unit: **word** which is 32 bits.

Salsa20 Stream Cipher

Notation: $\oplus$ is the usual bit-wise XOR. $+$ is mod 2^{32} addition. $<<<$ will mean you circular shift bits to the left.
Basic unit: **word** which is 32 bits.
Basic Operation: On input four words (a, b, c, d), $QR(a, b, c, d)$ is

Salsa20 Stream Cipher

Notation: $\oplus$ is the usual bit-wise XOR. $+$ is mod 2^{32} addition. $<<<$ will mean you circular shift bits to the left.
Basic unit: **word** which is 32 bits.
Basic Operation: On input four words (a, b, c, d), $QR(a, b, c, d)$ is

$$b := (b \oplus (b + d)) <<< 7$$

Salsa20 Stream Cipher

Notation: $\oplus$ is the usual bit-wise XOR. $+$ is mod 2^{32} addition.
$<<<$ will mean you circular shift bits to the left.
Basic unit: **word** which is 32 bits.
Basic Operation: On input four words (a, b, c, d), $QR(a, b, c, d)$ is

$$b := (b \oplus (b + d)) <<< 7$$

$$c := (c \oplus (a + b)) <<< 9$$

Salsa20 Stream Cipher

Notation: $\oplus$ is the usual bit-wise XOR. $+$ is mod 2^{32} addition. $<<<$ will mean you circular shift bits to the left.
Basic unit: **word** which is 32 bits.
Basic Operation: On input four words (a, b, c, d), $QR(a, b, c, d)$ is

$$b := (b \oplus (b + d)) <<< 7$$

$$c := (c \oplus (a + b)) <<< 9$$

$$d := (d \oplus (b + c)) <<< 13$$

Salsa20 Stream Cipher

Notation: $\oplus$ is the usual bit-wise XOR. $+$ is mod 2^{32} addition. $<<<$ will mean you circular shift bits to the left.
Basic unit: **word** which is 32 bits.
Basic Operation: On input four words (a, b, c, d), $QR(a, b, c, d)$ is

$$b := (b \oplus (b + d)) <<< 7$$

$$c := (c \oplus (a + b)) <<< 9$$

$$d := (d \oplus (b + c)) <<< 13$$

$$a := (a \oplus (c + d)) <<< 18$$

Salsa20 Stream Cipher

Notation: $\oplus$ is the usual bit-wise XOR. $+$ is mod 2^{32} addition. $<<<$ will mean you circular shift bits to the left.
Basic unit: **word** which is 32 bits.
Basic Operation: On input four words (a, b, c, d), $QR(a, b, c, d)$ is

$$b := (b \oplus (b + d)) <<< 7$$

$$c := (c \oplus (a + b)) <<< 9$$

$$d := (d \oplus (b + c)) <<< 13$$

$$a := (a \oplus (c + d)) <<< 18$$

Note: $\oplus$ and $+$ and $<<<$ are **fast!** So $QR(a, b, c, d)$ is **fast!** .

Salsa20 Stream Cipher

Notation: $\oplus$ is the usual bit-wise XOR. $+$ is mod 2^{32} addition. $<<<$ will mean you circular shift bits to the left.
Basic unit: **word** which is 32 bits.
Basic Operation: On input four words (a, b, c, d), $QR(a, b, c, d)$ is

$$b := (b \oplus (b + d)) <<< 7$$

$$c := (c \oplus (a + b)) <<< 9$$

$$d := (d \oplus (b + c)) <<< 13$$

$$a := (a \oplus (c + d)) <<< 18$$

Note: $\oplus$ and $+$ and $<<<$ are **fast!** So $QR(a, b, c, d)$ is **fast!** .

Note: Scrambles up a, b, c, d **a lot!** .

Salsa20 Stream Cipher-Init

Salsa20 Stream Cipher-Init

Initially have a 4×4 array of bytes (8 bits).

Const	Key	Key	Key
Key	Const	nonce	nonce
Pos	Pos	Const	Key
Key	Key	Key	Const

Salsa20 Stream Cipher-Init

Initially have a 4×4 array of bytes (8 bits).

Const	Key	Key	Key
Key	Const	nonce	nonce
Pos	Pos	Const	Key
Key	Key	Key	Const

View as 8 words by reading up-down, left-right

Salsa20 Stream Cipher-Init

Initially have a 4×4 array of bytes (8 bits).

Const	Key	Key	Key
Key	Const	nonce	nonce
Pos	Pos	Const	Key
Key	Key	Key	Const

View as 8 words by reading up-down, left-right

Const: Constants that are standardized. Public

Salsa20 Stream Cipher-Init

Initially have a 4×4 array of bytes (8 bits).

Const	Key	Key	Key
Key	Const	nonce	nonce
Pos	Pos	Const	Key
Key	Key	Key	Const

View as 8 words by reading up-down, left-right

Const: Constants that are standardized. Public

Key: Known only to Alice and Bob, used for long time. Private.

Salsa20 Stream Cipher-Init

Initially have a 4×4 array of bytes (8 bits).

Const	Key	Key	Key
Key	Const	nonce	nonce
Pos	Pos	Const	Key
Key	Key	Key	Const

View as 8 words by reading up-down, left-right

Const: Constants that are standardized. Public

Key: Known only to Alice and Bob, used for long time. Private.

Pos: These will start at 0 and increment every time used. Public.

Salsa20 Stream Cipher-Init

Initially have a 4×4 array of bytes (8 bits).

Const	Key	Key	Key
Key	Const	nonce	nonce
Pos	Pos	Const	Key
Key	Key	Key	Const

View as 8 words by reading up-down, left-right

Const: Constants that are standardized. Public

Key: Known only to Alice and Bob, used for long time. Private.

Pos: These will start at 0 and increment every time used. Public.

Salsa20 Stream Cipher-Init and other Issues

Initialize for R Rounds:

Even round do $QR(a, b, c, d)$ on the rows,

Salsa20 Stream Cipher-Init and other Issues

Initialize for R Rounds:
Even round do $QR(a, b, c, d)$ on the rows,
Every odd round do $QR(a, b, c, d)$ on the columns.

Salsa20 Stream Cipher-Init and other Issues

Initialize for R Rounds:
Even round do $QR(a, b, c, d)$ on the rows,
Every odd round do $QR(a, b, c, d)$ on the columns.

How Many Rounds: Salsa20 sets it to 20. Duh.

Salsa20 Stream Cipher-Init and other Issues

Initialize for R Rounds:
Even round do $QR(a, b, c, d)$ on the rows,
Every odd round do $QR(a, b, c, d)$ on the columns.

How Many Rounds: Salsa20 sets it to 20. Duh.

Salsa20 Stream Cipher-GetBits

We now have a well mixed 4×4 array of bytes (8 bits).

Salsa20 Stream Cipher-GetBits

We now have a well mixed 4×4 array of bytes (8 bits).
Could that just be our random bits? **Discuss**

Salsa20 Stream Cipher-GetBits

We now have a well mixed 4×4 array of bytes (8 bits). Could that just be our random bits? **Discuss**

No! All steps are reversible. From that array one can work backwards and find the Key!

Salsa20 Stream Cipher-GetBits

We now have a well mixed 4×4 array of bytes (8 bits).
Could that just be our random bits? **Discuss**

No! All steps are reversible. From that array one can work backwards and find the Key!
Just one more step:

Salsa20 Stream Cipher-GetBits

We now have a well mixed 4×4 array of bytes (8 bits).
Could that just be our random bits? **Discuss**

No! All steps are reversible. From that array one can work backwards and find the Key!
Just one more step:

Let the 4×4 array be $x[0], \ldots, x[15]$.

Salsa20 Stream Cipher-GetBits

We now have a well mixed 4×4 array of bytes (8 bits).
Could that just be our random bits? **Discuss**

No! All steps are reversible. From that array one can work backwards and find the Key!
Just one more step:

Let the 4×4 array be $x[0], \ldots, x[15]$.

Let the 4×4 initial array be $in[0], \ldots, in[15]$.

Salsa20 Stream Cipher-GetBits

We now have a well mixed 4×4 array of bytes (8 bits). Could that just be our random bits? **Discuss**

No! All steps are reversible. From that array one can work backwards and find the Key!
Just one more step:

Let the 4×4 array be $x[0], \dots, x[15]$.

Let the 4×4 initial array be $in[0], \dots, in[15]$.

For $i = 0$ to 15 output $x[i] + in[i]$.

Salsa20 Stream Cipher-GetBits

We now have a well mixed 4×4 array of bytes (8 bits). Could that just be our random bits? **Discuss**

No! All steps are reversible. From that array one can work backwards and find the Key!
Just one more step:

Let the 4×4 array be $x[0], \ldots, x[15]$.

Let the 4×4 initial array be $in[0], \ldots, in[15]$.

For $i = 0$ to 15 output $x[i] + in[i]$.

Security: Salsa20 was introduced in 2005 and has not been broken. See Wikipedia page for partial attacks (e.g., Salsa8).

How to Design a Good Stream Cipher?

SC's are designed, used, and not broken

How to Design a Good Stream Cipher?

SC's are designed, used, and not broken until they are.

How to Design a Good Stream Cipher?

SC's are designed, used, and not broken until they are.

Frustrating: Can prove a Stream Cipher is BAD but not GOOD.

How to Design a Good Stream Cipher?

SC's are designed, used, and not broken until they are.

Frustrating: Can prove a Stream Cipher is BAD but not GOOD.

Jon Katz:

> *Absent proofs, the only ways to claim that a stream cipher is good are to (1) follow known design principles and (2) make sure known attacks do not work. It helps lend credibility if they are designed by people who know what they are doing, not just throwing random stuff together, but I realize that's not very scientific.*

How to Design a Good Stream Cipher?

SC's are designed, used, and not broken until they are.

Frustrating: Can prove a Stream Cipher is BAD but not GOOD.

Jon Katz:

> *Absent proofs, the only ways to claim that a stream cipher is good are to (1) follow known design principles and (2) make sure known attacks do not work. It helps lend credibility if they are designed by people who know what they are doing, not just throwing random stuff together, but I realize that's not very scientific.*
>
> *Trivium, in particular, always struck me as so simple that it cannot possibly be secure. And yet, there are no attacks. But I don't think it has been subject to the same scrutiny as AES, or even RC4. ChaCha is actually used, so people care about its security. Hence its security seems solid. For now.*

Good Science and Bad Science

Karl Popper (1930's): A Scientific Theory should be **falsifiable** . Propose experiments that could show it is **not** true. The longer the theory survives scrutiny the more likely it is to be true.

Good Science and Bad Science

Karl Popper (1930's): A Scientific Theory should be **falsifiable** . Propose experiments that could show it is **not** true. The longer the theory survives scrutiny the more likely it is to be true.

1) **Classical Mechanics:** Good Science. Many experiments proposed and carried out. Confirmed it until had problems with fast speeds and small particles.

Good Science and Bad Science

Karl Popper (1930's): A Scientific Theory should be **falsifiable** . Propose experiments that could show it is **not** true. The longer the theory survives scrutiny the more likely it is to be true.

1) **Classical Mechanics:** Good Science. Many experiments proposed and carried out. Confirmed it until had problems with fast speeds and small particles.

2) **Quantum Mechanics:** Good Science. Many experiments proposed and carried out. So far has not been falsified. Yet.

Good Science and Bad Science

Karl Popper (1930's): A Scientific Theory should be **falsifiable** . Propose experiments that could show it is **not** true. The longer the theory survives scrutiny the more likely it is to be true.

1) **Classical Mechanics:** Good Science. Many experiments proposed and carried out. Confirmed it until had problems with fast speeds and small particles.

2) **Quantum Mechanics:** Good Science. Many experiments proposed and carried out. So far has not been falsified. Yet.

3) **Libertarianism Theory:** Bad Science:
Everything bad is the governments fault w/o looking at data. Global warming require government action, hence its false.

Good Science and Bad Science

Karl Popper (1930's): A Scientific Theory should be **falsifiable** . Propose experiments that could show it is **not** true. The longer the theory survives scrutiny the more likely it is to be true.

1) **Classical Mechanics:** Good Science. Many experiments proposed and carried out. Confirmed it until had problems with fast speeds and small particles.

2) **Quantum Mechanics:** Good Science. Many experiments proposed and carried out. So far has not been falsified. Yet.

3) **Libertarianism Theory:** Bad Science:
Everything bad is the governments fault w/o looking at data.
Global warming require government action, hence its false.

4) **Communism:** Bad Science:
Wages go down – Capitalists exploiting the worker.
Wages to up – Capitalists placating the worker to avoid revolution.

Good Crypto and Bad Crypto

A Scientific Theory should be **falsifiable** . Propose experiments that could show it is **not** true. The longer the theory survives scrutiny the more likely it is to be true. For now.

Good Crypto and Bad Crypto

A Scientific Theory should be **falsifiable** . Propose experiments that could show it is **not** true. The longer the theory survives scrutiny the more likely it is to be true. For now.

An encryption system should be **falsifiable** . Propose ways to break it. The longer it stays unbroken the more likely it is to be unbreakable. For now. **Caveat:** let many people try! Kerchoffs's law very useful here!

Good Crypto and Bad Crypto

A Scientific Theory should be **falsifiable** . Propose experiments that could show it is **not** true. The longer the theory survives scrutiny the more likely it is to be true. For now.

An encryption system should be **falsifiable** . Propose ways to break it. The longer it stays unbroken the more likely it is to be unbreakable. For now. **Caveat:** let many people try! Kerchoffs's law very useful here!

Speculation: Does the NSA let outsiders try to break their systems? If not then might not be Good Crypto. I really do not know.

Good Crypto and Bad Crypto

A Scientific Theory should be **falsifiable** . Propose experiments that could show it is **not** true. The longer the theory survives scrutiny the more likely it is to be true. For now.

An encryption system should be **falsifiable** . Propose ways to break it. The longer it stays unbroken the more likely it is to be unbreakable. For now. **Caveat:** let many people try! Kerchoffs's law very useful here!

Speculation: Does the NSA let outsiders try to break their systems? If not then might not be Good Crypto. I really do not know.
I tried asking them but they wouldn't tell me!