Stream ciphers

Rivest Cipher 4 (RC4)

- Designed in 1987
- Designed to have good performance in software, rather than hardware
- *No longer considered secure*, but still interesting to study
  - Simple description; not LFSR-based
  - Still encountered in practice
  - Interesting attacks

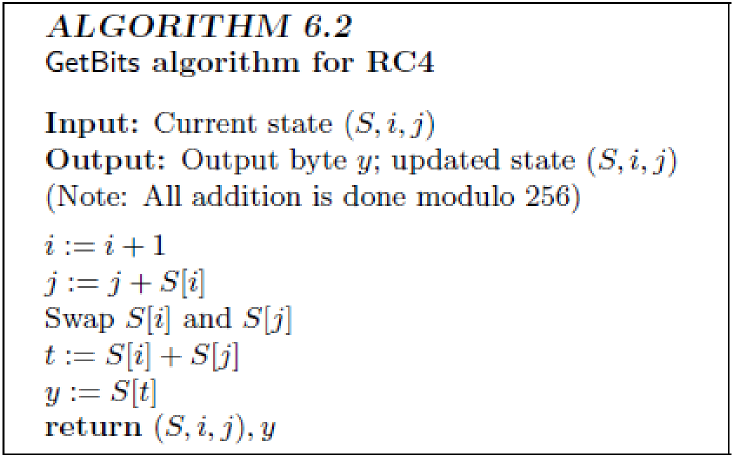
RC4-Init

Init for RC4. Addition is mod 256.

```
1. 16 byte Key $k[0], \ldots, k[15]$
2. For $i = 0$ to 255
    2.1 $S[i] := i$. $S$ is 256 bytes.
    2.2 $k[i] := k[i \bmod 16]$. $k$ is now 256 bytes.
3. $j := 0$
4. For $i = 0$ to 255
    4.1 $j := j + S[i] + k[i]$
    4.2 Swap $S[i]$ and $S[j]$
5. $i := 0$, $j := 0$, Return $(S, i, j)$.
```

Note:
1) RC4 deals with bytes, not bits.
2) This is just Init Stage, no output byte yet
3) Can use an IV by prepending IV to Key and using that as Key

RC4-GetBits



ALGORITHM 6.2
GetBits algorithm for RC4

Input: Current state (S, i, j)
Output: Output byte y; updated state (S, i, j)
(Note: All addition is done modulo 256)

```
$i := i + 1$
$j := j + S[i]$
Swap $S[i]$ and $S[j]$
$t := S[i] + S[j]$
$y := S[t]$
return $(S, i, j), y$
```

Note: Even though called GetBits, actually gets bytes.

RC4

- Not designed to take an IV, but often used with an IV anyway
  - E.g. prepend IV to the key

Attack 1: bias in 2nd output byte

- Let S_t denote permutation after t steps
  - Treat S_0 as uniform for simplicity
- Say $X = s_0[1] \neq 2$ and $S_0[2] = 0$
  - Occurs with probability $\frac{1}{256}$
  - Then:
    - After 1 step, $S_1[X] = X, i = 1, j = X$
    - After 2 steps, $j = x$, output $S_2[X] = 0$
- Otherwise,$S_2[X]$ is uniform
  - Overall probability of 2nd byte 0 is about $\frac{2}{256}$

Attack 1 Continued: bias in 2nd output byte

If uniform then $\Pr(\text{2nd byte is 0}) = \frac{1}{256}$.

We have seen that $\Pr(\text{2nd byte is 0}) = \frac{2}{256}$.

Is this a problem?

Attack 1 Continued: bias in 2nd output byte

If uniform then $\Pr(\text{2nd byte is 0}) = \frac{1}{256}$.

We have seen that $\Pr(\text{2nd byte is 0}) = \frac{2}{256}$.

Is this a problem? Yes!

1. If Eve assumes 2nd byte is 0 and uses that to get more information she will be correct more often then we would like.
2. Exposes general structural problem with RC4.

Attack 2: From first 3 bytes can get 4th w/prob $\frac{5}{100}$

Details of calculations mentioned here are in online notes.

Example: $k[0] = 3$, $k[1] = 255$, $k[2] = X$ (known).

Can show that with prob 5% after init phase:

1. $S[0] = 3$
2. $S[1] = 0$
3. $S[3] = X + 6 + k[3]$

Can show that given the above $y_1 = S[3] = X + 6 + k[3]$.

Since Eve knows X can find $k[3]$

Attack 2: From first 3 bytes can get 4th w/prob $\frac{5}{100}$

Details of calculations mentioned here are in online notes.

Example: $k[0] = 3$, $k[1] = 255$, $k[2] = X$ (known).

Can show that with prob 5% after init phase:

1. $S[0] = 3$
2. $S[1] = 0$
3. $S[3] = X + 6 + k[3]$

Can show that given the above $y_1 = S[3] = X + 6 + k[3]$.

Since Eve knows X can find $k[3]$ with prob $\frac{5}{100}$.
Caveat: Eve won't know when she's right.

Attack 2: When Is Last Slide Useful to Eve?

Scenario: Alice and Bob use the same key over and over by also a 3-byte IV that they change a lot. Good idea?

Attack 2: When Is Last Slide Useful to Eve?

Scenario: Alice and Bob use the same key over and over by also a 3-byte IV that they change a lot. Good idea? NO!
Alice and Bob are using $IV[0], IV[1], IV[2], k[0]$
Note: Eve knows $IV[0], IV[1], IV[2]$ so effectively does know first three bytes of a key.
Eve:

1. For each 3-byte IV Alice and Bob use Eve determines if it will give her $> \frac{1}{256}$ prob to deduce fourth byte.
2. If so then use it to deduce fourth byte even though prob isn't that high.
3. Keep all of that data! Some byte will end up being much more likely than the others. Thats the one!

Attack 2: Finishing it up

1. Can extend to first t bytes helps get $t + 1$st byte.
2. Can iterate until get entire key.
3. Takes time, patience, and an Alice and Bob who use the same key for a while.
4. RC4 no longer considered secure.

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. Metal Education. Click HERE for great deal on Tuxedos!

First Comment on Blog

Great post on Trivial! Hardware Cube Attack. Metal Education. Click HERE for great deal on Tuxedos!

My Response

First Comment on Blog

Great post on Trivial! Hardware Cube Attack. Metal Education. Click HERE for great deal on Tuxedos!

My Response
No response. However,

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

Too bad. They called it a Great post. Oh well.

Second Comment on Blog

meh squared

Second Comment on Blog

meh squared

My response

Second Comment on Blog

meh squared

My response

I take it you are not impressed with Trivium. Can you say why so you can enlighten by readers and I can enlighten my class?

Third 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.

If you are interested in teaching useful (and used) stream ciphers, you could start with RC4, which was widely used in TLS (i.e. encrypting a lot Internet traffic) until it was very badly broken. RC4 exhibits all sorts of interesting weaknesses for teaching, and it is very simple.

My understanding is that the most widely used stream cipher will soon likely be Chacha20 (again for TLS). The authentication mechanism (Poly1305) and other Wegman-Carter-type MACs involve some algebra and probability that are interesting for teaching crypto as well.

Salsa20 Stream Cipher (not to be confused with Nelson's Salsa Class)

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

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.

Nonce: This IV but can only use a string once. Public

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

Note: Nonce: Number-used-once. Public here but not necc

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.

Nonce: How to gen rand-looking strings w/o repeats? Discuss

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.

Nonce: How to gen rand-looking strings w/o repeats? Discuss

1) Store all prior strings. Too much space.

2) Concat year-month-day-time and a random string. Works!

3) There are other ways.

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:

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. 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.

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.

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!