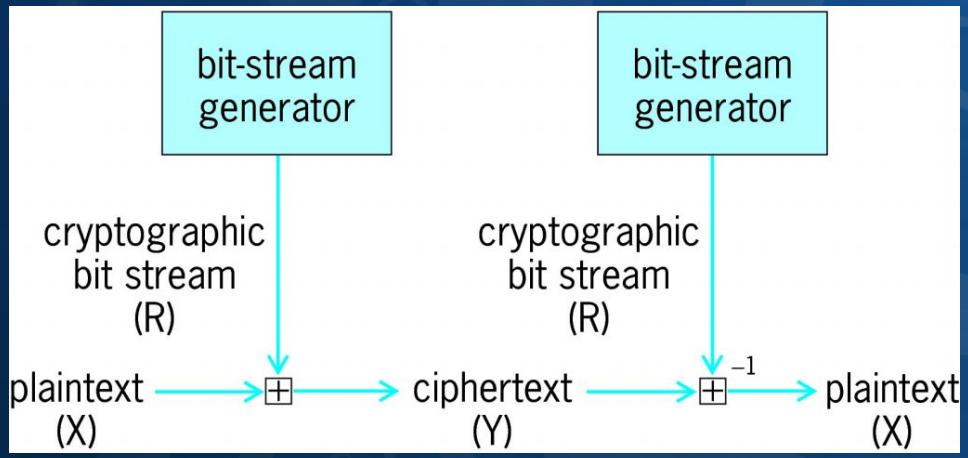
Exploring non-combiner type stream ciphers

Ivan Voras, <ivoras@gmail.com>

What are "combiner-type stream ciphers"?

- The majority of stream ciphers, either designed directly (e.g. RC4) or adapted from block ciphers (e.g. AES-CTR)
- The most important property: they generate a keystream using a PRNG algorithm; this keystream is XOR-ed with the plaintext to produce the ciphertext



“Stream ciphers are evil”

- They are not… if used correctly
- The heart of the problem: the XOR step
  - **Reusing the keystream twice will reveal the keystream**

A XOR K = U
B XOR K = V

U XOR V = K

The WiFi WEP fiasco…

- 802.11 specified WEP: “wired-equivalent privacy” using RC4
  - RC4 is *very fast* and *very easy*
  - But it is a combiner-type stream cipher
- The keystream is initialized with a 24-bit IV, making it 50% likely to repeat after only 5000 packets
  - A passive attack in 3 minutes!
- There are also other problems in WEP…

His excellence, the king: RC4

```
1.  static u_char inline
2.  rc4_next(struct rc4_state *const state)
3.  {
4.          u_char j;
5.
6.          /* Update modification indicies */
7.          state->index1++;
8.          state->index2 += state->perm[state->index1];
9.
10.         /* Modify permutation */
11.         swap_bytes(&state->perm[state->index1], &state->perm[state->index2]);
12.
13.         /* Encrypt/decrypt next byte */
14.         j = state->perm[state->index1] + state->perm[state->index2];
15.         return state->perm[j];
16. }
17.
```

- … 5 lines of active code … 300 MB/s [asm]
- Hacking RC4 is like fishing in a barrel - with dynamite - and just as dangerous

Why use stream ciphers at all?

- Stream ciphers are more convenient than block ciphers:
  - There is no need to divide data into blocks (introducing padding, etc.)
  - There are no "modes" to consider
  - There are (ideally) no explicit IVs
- Traditionally, block ciphers get much more research and scrutiny
  - We can build stream ciphers from block ciphers but not the other way around!

The CTR mode: the easy way

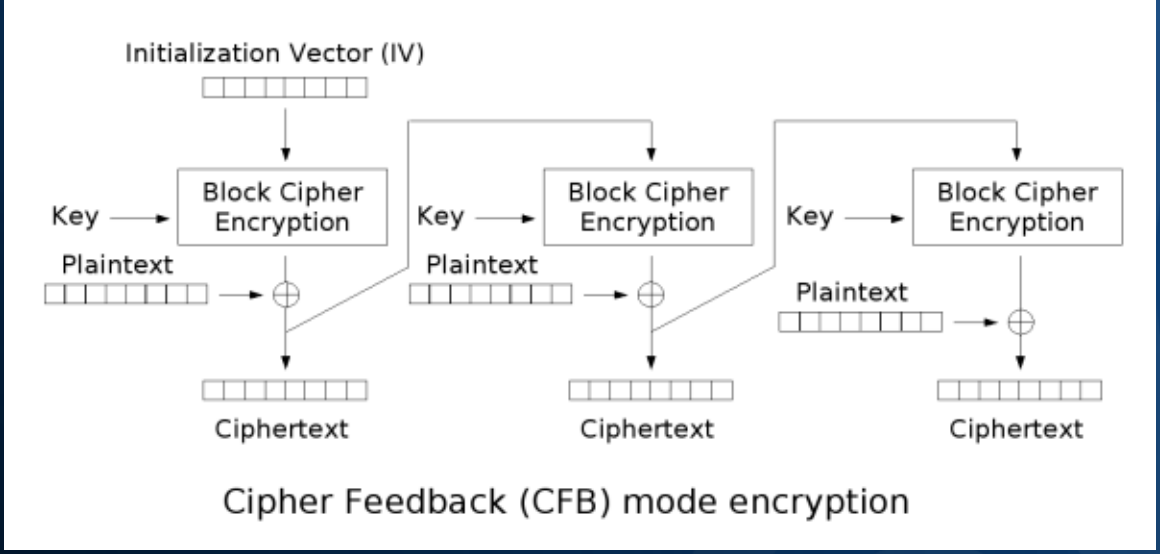
- CTR is a “counter”
- Steps:
  - Start with an IV
    - Encrypt the IV with K
    - Each byte of the result forms a keystream
    - Use XOR…
  - Increment IV (IV += 1)
- Each block encryption oper. (e.g. AES) results in 16 bytes of a stream cipher
  - Combiner-type stream cipher

The IV becomes the "weakest link"

It "distinguishes" keystreams

The CFB and OFB modes: much better

- "Cipher / Output FeedBack" modes
  - **Still XOR-based**
  - **Make the resulting keystream dependant on the previous data** (sort of similar guarantees as CBC)



Cipher Feedback (CFB) mode encryption

Can we do better?

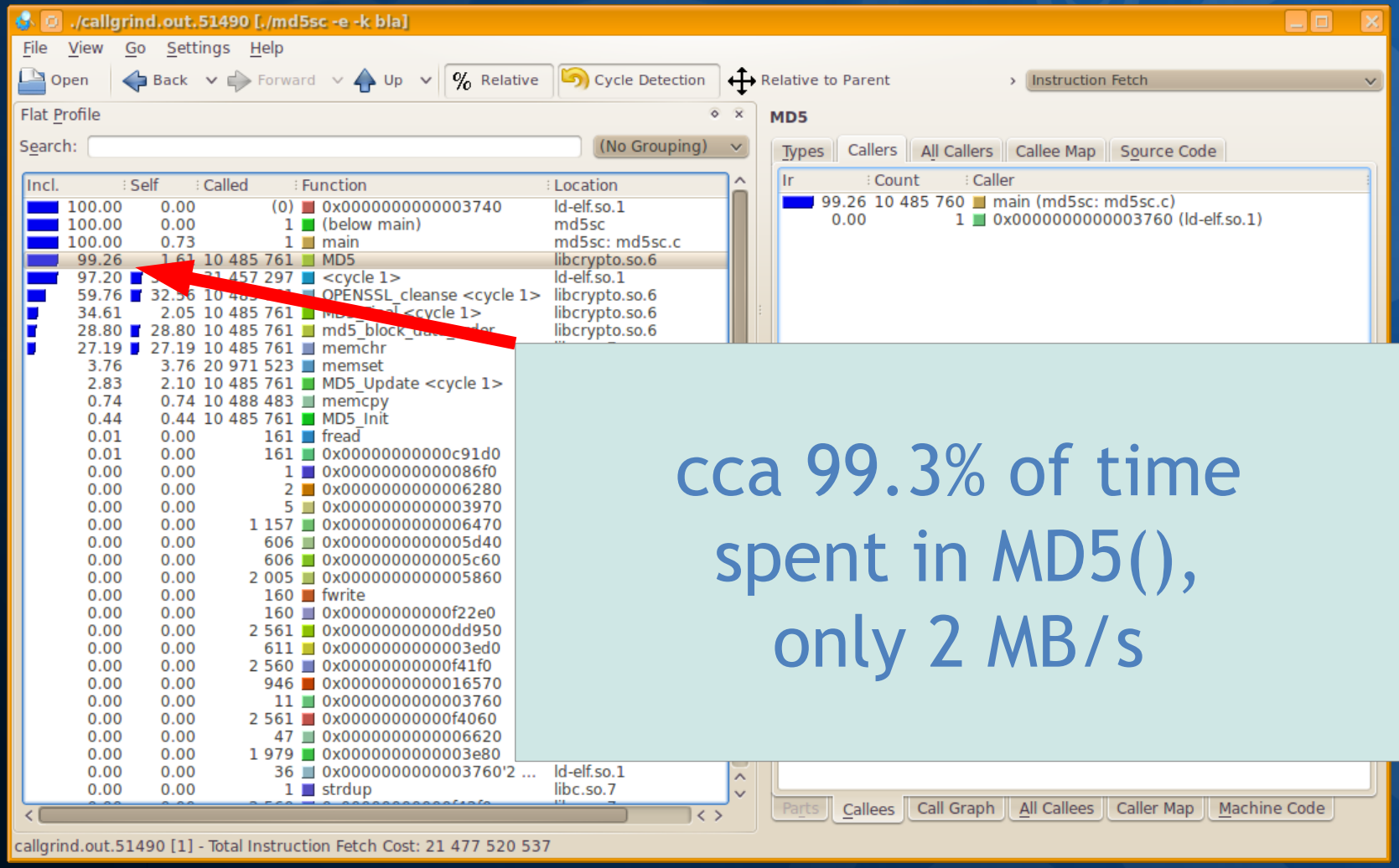
- Note: brainstorming ahead, not peer-reviewed!

- Some ideas:
  - Use a one way function to mix data with a PRNG
  - Try adapting common algorithms
  - How complex / cheap can this function be?

A variant with MD5

- Hash the password into a 16-byte buffer
- For each byte of plaintext:
  - out_byte = in_byte XOR md5buf[0]
  - md5buf[15] = out_byte
  - md5buf = MD5(md5buf)
- Elegant structure…
- XOR keystream reuse weakness avoided
- Only it turns out MD5() is slow… 2 MB/s
  - …and MD4 is only 20% faster…

A variant with MD5



cca 99.3% of time spent in MD5(), only 2 MB/s

Can we do it cheaper?

- Not really.
- The “mixing” quality of a hash function is very important
  - MD5 is actually very good – there are no known problems with this approach
- Using adler32() instead of MD5():
  - 30 MB/s (15 times faster!)
  - The output is *visibly* distinguishable from random – completely unusable

But just for laughs...

- We need a good and cheap mixer
  - How about AES? Too slow...
  - BUT, how about Rijndael?
- Two rounds of Rijndael
  - ~~ 22 MB/s
  - Security apparently good – no obvious problems (after 10 bytes you get full AES)
- MD5 with fewer rounds?
  - Doesn't seem a good idea – different transformations in each of 4 rounds

Cue distinction between Rijndael and AES

The "RC4-skip"

- Idea #7123: The RC4-skip
- For each input byte *N*:
  - **Encrypt it with the next calculated keystream byte**
  - **advance the keystream P2[*N*] times**
- Pros: the structure of RC4 is intact
- Cons: we slowed RC4 down 128 times
  - **Want it cheaper? Weaken the P2 table (skip table)… with 0-15 skips, 33 MB/s**

The "RC4 P-nuke"

- "Ok, advancing the stream is too slow, what else can we do?"
- For each byte:
  - Encrypt it with the next calculated keystream byte
  - XOR the entire P-box with the byte
- Math says the XOR-ed P-box remains a P-box
  - XOR is one of the fastest HW ops
  - 29 MB/s … still slow
    - Must XOR 256 bytes for every incoming byte

Summary

Algorithm	Speed	Safe?
RC4 (plain)	150 MB/s	Yes
AES-CBC (plain)	80 MB/s	Yes
MD5 stream	2.1 MB/s	Yes
MD4 stream	2.4 MB/s	Yes
Adler32 stream	34 MB/s	No
Rijndael 2r stream	22 MB/s	Probably
RC4-skip	33 MB/s	Yes
RC4 P-nuke	29 MB/s	Probably

- From performance point of view, a failure
- … but an interesting failure

What next?

- The search for a good fast mixer function continues...
- RC4 is always fun to hack
- Source code for these algorithms available at: http://goo.gl/zy4IF

Thank you!

Ivan Voras <ivoras@gmail.com>