All Your Biases Belong To Us: Breaking RC4 in WPA-TKIP and TLS

Mathy Vanhoef and Frank Piessens, KU Leuven
USENIX Security 2015 (best student paper)

Presentation for USENIC ATC ‘16
RC4

Intriguingly simple stream cipher

~ 10 lines in Python

WEP
WPA-TKIP

SSL / TLS

PPP/MPPE

And others ...
Intriguingly simple stream cipher

~ 10 lines in Python
Why study RC4?

Immune to recent attacks on SSL/TLS:

- 2003: Padding oracle
- 2011: BEAST
- 2013: Lucky 13
- 2014: POODLE

Solution: use stream cipher or up-to-date TLS library

Only widely supported option was RC4
RC4 was heavily used!

ICSI Notary: #TLS connections using RC4
Browser support today (June 2016)

- Chrome: dropped support in v48 (20 Jan. 2016)
- Firefox: dropped support in v44 (26 Jan. 2016)
- IE11: supports RC4
- Edge: supports RC4

“will be disabled in forthcoming update”
Contributions: why RC4 must die

New Biases

\[ \lambda_{\hat{\mu}} = (1 - \alpha(g))^{|C| - |\hat{\mu}|} \cdot \alpha(g)^{|\hat{\mu}|} \]

Plaintext Recovery

Break WPA-TKIP

Attack HTTPS
Contributions: why RC4 must die

New Biases

\[ \lambda_{\hat{\mu}} = (1 - \alpha(g)) |c - \hat{\mu}| \cdot \alpha(g) |\hat{\mu}| \]

Plaintext Recovery

Break WPA-TKIP

Attack HTTPS
First: Existing Biases

We want a straight line.

\[ \Pr[Z_2 = 0] = \frac{2}{256} \quad [\text{MS01}] \]
First: Existing Biases

Distribution keystream byte 1 (to 256)

AlFardan et al. ‘13: first 256 bytes biased

We want a straight line..
Long-Term Biases

Fluhrer-McGrew (2000):
- Some consecutive values are biased
  
  Examples: (0, 0) and (0, 1)

Mantin’s ABSAB Bias (2005):
- A byte pair \((A, B)\) likely reappears
Search for new biases

Traditional emperical approach:

▪ Generate large amount of keystreams
▪ Manually inspect data or graph

Fluhrer-McGrew biases: only 8 of 65 536 pairs are biased

How to automate the search?
Search for new biases

Traditional empirical approach:
- Generate large amount of keystreams
- Manually inspect data or graph

Hypothesis tests!
- Uniformly distributed: Chi-squared test.
- Correlated: M-test (detect outliers = biases)

→ Allows a large-scale search, revealing many new biases
Example: keystream byte 258
Biases in Bytes 258-513

Example: keystream byte 320
Example: keystream byte 352

Biases quickly become quite weak
New Long-term Bias

\((Z_{256 \cdot w}, Z_{256 \cdot w+2}) = (0, 128)\)

with probability \(2^{-16}(1 + 2^{-8})\)

Every block of 256 bytes
Additional Biases

See paper!
Contributions: why RC4 must die

New Biases

Break WPA-TKIP

Plaintext Recovery

Attack HTTPS

\[ \lambda_{\hat{u}} = (1 - \alpha(g))^{|c| - |\hat{u}|} \cdot \alpha(g)^{|\hat{u}|} \]
Existing Methods [AlFardan et al. ‘13]

Plaintext encrypted under several keystreams

Ciphertext Distribution $\bullet$ Plaintext guess $\mu$ $\Rightarrow$ Induced keystream distribution

Verify guess: how close to real keystream distribution?
Example: Decrypt byte 1

Ciphertext Distribution
Example: Decrypt byte 1

RC4 & Ciphertext distribution

![Graph showing RC4 & Ciphertext distribution with a line plot and markers indicating data points. The x-axis is labeled with numbers from 0 to 256, and the y-axis ranges from 0.003879 to 0.003913879. The plot includes a significant drop around the 192 mark.]
Example: Decrypt byte 1

If plaintext byte $\mu = 0x28$: **RC4 & Induced**

$\mu = 0x28$ has **low likelihood**
Example: Decrypt byte 1

If plaintext byte $\mu = 0x5C$: **RC4 & Induced**

$\mu = 0x5C$ has **higher likelihood**
Example: Decrypt byte 1

If plaintext byte $\mu = 0x5A$: RC4 & Induced

$\mu = 0x5A$ has highest likelihood!
Types of likelihood estimates

Previous works: pick value with highest likelihood.

Better idea: list of candidates in decreasing likelihood:
- Most likely one may not be correct!
- Prune bad candidates (e.g. bad CRC)
- Brute force cookies or passwords

How to calculate list of candidates?
1st idea: Generate List of Candidates

Gist of the Algorithm: Incremental approach

Calculate candidates of length 1, length 2, ...
2nd idea: abusing the ABSAB bias

Assume there’s **surrounding known plaintext**!

- Derive values of \((A, B)\)
- Combine with ABSAB bias to (probabilistically) predict \((A', B')\)
- Ordinary likelihood calculation over only \((A', B')\)

Likelihood estimate: 

\[
\lambda_{\hat{\mu}} = (1 - \alpha(g))|c| - |\hat{\mu}| \cdot \alpha(g)|\hat{\mu}|
\]
Contributions: why RC4 must die

New Biases

Plaintext Recovery

Break WPA-TKIP

Attack HTTPS

$$\lambda_{\hat{\mu}} = (1 - \alpha(g)) |c - |\hat{\mu}| \cdot \alpha(g) |\hat{\mu}|$$
How are packets sent/received?

<table>
<thead>
<tr>
<th>IV</th>
<th>Data</th>
<th>MIC</th>
<th>CRC</th>
</tr>
</thead>
</table>

Encrypted
TKIP Background

How are packets sent/received?

1. Add Message Integrity Check (MIC)
TKIP Background

How are packets sent/received?

1. Add Message Integrity Check (MIC)
2. Add CRC (leftover from WEP)
TKIP Background

How are packets sent/received?

1. Add Message Integrity Check (**MIC**)
2. Add **CRC** (leftover from WEP)
3. Add **IV** (increments every frame)
TKIP Background

How are packets sent/received?

1. Add Message Integrity Check (MIC)
2. Add CRC (leftover from WEP)
3. Add IV (increments every frame)
4. Encrypt using RC4 (per-packet key)
Flaw #1: TKIP Per-packet Key

(Key, IV) -> per-packet key
Flaw #1: TKIP Per-packet Key

\[ (IV_0, IV_1) \rightarrow \text{Anti-FMS} \]

\( \rightarrow IV \)-dependent biases in keystream

[Gupta/Paterson et al.]
Flaw #2: MIC is invertible

If decrypted, reveals MIC key

→ With the MIC key, an attacker can inject and decrypt some packets [AsiaCCS ‘13]
Goal: decrypt data and MIC

If decrypted, reveals MIC key

Goal: decrypt packet using RC4 biases & derive MIC key
  - Problem: must generate many identical WPA-TKIP packets
  - Solution: make victim connect to our server and retransmit identical TCP packets

Generate list of packet candidates
  - Prune bad candidates based on CRC
Evaluation

~ 8 million encryptions of packet

Ciphertext copies times $2^{20}$
Evaluation

- ~ 8 million encryptions of packet
- Takes 1 hour with 2500 packets / second

Graph showing the probability of key recovery with different numbers of candidates and ciphertext copies times $2^{20}$.
Contributions: why RC4 must die

- New Biases

- Plaintext Recovery

\[ \lambda_{\hat{\mu}} = (1 - \alpha(g)) |c - |\hat{\mu}| \cdot \alpha(g)|\hat{\mu}| \]

- Break WPA-TKIP

- Attack HTTPS
TLS Background

→ Focus on record protocol with RC4 as cipher
Targeting HTTPS Cookies

Previous attacks only used Fluhrer-McGrew (FM) biases.

We combine FM biases and ABSAB biases.

To use ABSAB biases we first surround cookie with known data.

1. Remove unknown plaintext arround cookie.
2. Inject known plaintext arround cookie.
Example: manipulated HTTP request

User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko
Host: a.site.com
Connection: Keep-Alive
Cache-Control: no-cache
Cookie: auth=????????????????; P=aaaaaaaaaaaaaaaaaaaaa

Headers are predictable

Surrounded by known plaintext at both sides
Preparation: manipulating cookies

a.site.com  Client  fake.site.com

HTTPS  insecure

Remove & inject secure cookies!
Performing the attack!

STEP 1
Attacker injects code to generate requests

JavaScript: Cross-Origin requests in WebWorkers
Performing the attack!

**STEP 1**
Attacker injects code to generate requests

**STEP 2**
Attacker captures the encrypted requests

Keep-Alive connection to generate them fast
Performing the attack!

**STEP 1**
Attacker injects code to generate requests

**STEP 2**
Attacker captures the encrypted requests

**STEP 3**
Attacker computes likely cookies and tries each one

Combine Fluhrer-McGrew and ABSAB biases
Decryption of a 16-character cookie involves one billion encryptions of the cookie.

Graph showing the percentage of candidates for decryption, comparing 2^{23} candidates to 1 candidate. The graph indicates that 9 ciphertext copies times 2^{27} are equivalent to one billion encryptions.
Deciphering 16-character cookie

\[ 2^{23} \text{ candidates} \]
\[ 1 \text{ candidate} \]

\[ \sim \text{one billion encryptions of cookie} \]

Takes 75 hours with 4450 requests / second

Ciphertext copies times \[ 2^{27} \]
Decrypting 16-character cookie

DEMO!

rc4nomore.com
Questions?

May the bias be ever in your favor
Questions?

May the bias be ever in your favor
Shuffles permutation of $[0..255]$

<table>
<thead>
<tr>
<th></th>
<th>77</th>
<th>37</th>
<th>102</th>
<th>233</th>
<th>...</th>
<th>151</th>
<th>14</th>
<th>198</th>
<th>0</th>
<th>56</th>
</tr>
</thead>
</table>

**Secret index $j$**
- pseudo-randomly updated value

**Public index $i$**
- $= \text{keystream position mod } 256$

$\rightarrow$ Output byte selected based on index $j$ and $i$
Fallback to RC4

Client

ClientHello: without RC4

Server

ServerHello: use AES

Browser first tries without RC4
Alert: Handshake Failed

ClientHello: without RC4

Browser first tries without RC4

If that fails …
ClientHello: without RC4

Alert: Handshake Failed

ClientHello: with RC4

ServerHello: use RC4

Server

Browser first tries without RC4

If that fails ...

… fallback to RC4

Client

Fallback to RC4
Client Hello: without RC4

Alert: Handshake Failed

Client Hello: with RC4

Server Hello: use RC4

Browser first tries without RC4

Forgeable by attacker!

... fallback to RC4

➢ Fallback provides no security
➢ But useful to determine how many servers require RC4
Biases in Bytes 257-513

Distribution keystream byte 513

P-value \approx 10^{-300}

2^{47} keystreams
Additional Biases

Short-Term:
- $Z_1$ and $Z_2$ influence initial 256 bytes
- Consecutive bytes likely (in)equal

Long-term Biases:
- Byte value “likely” reappears

See paper!
Keystream bytes $Z_1$ and $Z_2$

$Z_1$ and $Z_2$ influence all initial 256 bytes

- $Z_1 = 257 - i \rightarrow Z_i = 0$
Keystream bytes $Z_1$ and $Z_2$

$Z_1$ and $Z_2$ influence all initial 256 bytes

- $Z_1 = 257 - i \Rightarrow Z_i = 0$
- $Z_1 = 257 - i \Rightarrow Z_i = i$
Keystream bytes $Z_1$ and $Z_2$

$Z_1$ and $Z_2$ influence all initial 256 bytes

- $Z_1 = 257 - i \rightarrow Z_i = 0$
- $Z_1 = 257 - i \rightarrow Z_i = i$
- $Z_2 = 0 \rightarrow Z_i \neq i$
Keystream bytes $Z_1$ and $Z_2$

$Z_1$ and $Z_2$ influence all initial 256 bytes

- $Z_1 = 257 - i \rightarrow Z_i = 0$
- $Z_1 = 257 - i \rightarrow Z_i = i$
- $Z_2 = 0 \rightarrow Z_i \neq i$
- And others

| 1 | 2 | ... | ≠ 50 | ≠ 51 | ...
|---|---|-----|------|------|-----
Example: Decrypt byte 1

If plaintext byte $\mu = 0x28$: **RC4 & Induced**

Likelihood of $\mu = \text{probability of witnessing induced, given the RC4 distribution}$
Example: Decrypt byte 1

If plaintext byte $\mu = 0x5C$: **RC4 & Induced**

Likelihood of $\mu = \text{probability of witnessing induced, given the RC4 distribution}$
Example: Decrypt byte 1

If plaintext byte $\mu = 0x5A$: **RC4 & Induced**

Likelihood of $\mu = $ probability of witnessing induced, given the RC4 distribution
Evaluation

Simulations with $2^{30}$ candidates:
- Need $\approx 2^{24}$ captures to decrypt with high success rates

Empirical tests:
- Server can inject 2 500 packets per second
- Roughly one hour to capture sufficient traffic
- Successfully decrypted packet & found MIC key!