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

Mathy Vanhoef and Frank Piessens, KU Leuven
USENIX Security 2015
Intriguingly simple stream cipher

WEP
WPA-TKIP
SSL / TLS
PPP/MPPE

And others ...
RC4

Intriguingly simple stream cipher

Key

RC4  →  Keystream  ◦  Plaintext  =  Ciphertext
Is RC4 still used?!

ICSI Notary: TLS connections using RC4

RC4 fallback not taken into account!
RC4 Fallback

Client

 Browser first tries without RC4

ClientHello: without RC4

ServerHello: use AES

Server
RC4 Fallback

Client

- ClientHello: without RC4
- Alert: Handshake Failed

Server

- Browser first tries without RC4
- If that fails …
Client

ClientHello: without RC4
Alert: Handshake Failed
ClientHello: with RC4
ServerHello: use RC4

Server

Browser first tries without RC4
If that fails …
… fallback to RC4
RC4 Fallback

Client

ClientHello: without RC4
Alert: Handshake Failed
ClientHello: with RC4
ServerHello: use RC4

Server

Browser first tries without RC4
Forgeable by attacker!
… fallback to RC4

➢ 13% estimate is a lower bound
➢ Force connection (which we assumed secure) to use RC4
Our Goal: further kill RC4

New Biases

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

Plaintext Recovery

Break WPA-TKIP

Attack HTTPS
First: Existing Biases

Distribution keystream byte 2

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

Distribution keystream byte 1
First: Existing Biases

Distribution keystream byte 1 (to 256)

AlFardan et al. ‘13: first 256 bytes biased
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 empirical approach:
- Generate large amount of keystreams
- Manually inspect data or graph

Fluhrer-McGrew: only 8 out 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
Biases in Bytes 258-513

Example: keystream byte 258

![Graph showing probability distribution for keystream byte value](image_url)
Example: keystream byte 320
Biases in Bytes 258-513

Example: keystream byte 352

Biases quickly become quite weak
New Long-term Bias

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

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

<table>
<thead>
<tr>
<th>128</th>
<th>0</th>
<th>...</th>
</tr>
</thead>
</table>

Every block of 256 bytes
Additional Biases

See paper!
Our Goal: further kill RC4

New Biases

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

Plaintext Recovery

Break WPA-TKIP

Attack HTTPS
Existing Methods [AlFardan et al. ‘13]

Plaintext encrypted under several keystreams

Verify guess: how close to real keystream distribution?

Ciphertext Distribution → Plaintext guess \( \mu \) → *Induced* keystream distribution
Example: Decrypt byte 1

Ciphertext Distribution
Example: Decrypt byte 1

RC4 & Ciphertext distribution
Example: Decrypt byte 1

If plaintext byte $\mu = 0x28$: \textbf{RC4 & Induced}

$\mu = 0x28$ has \textbf{low likelihood}
Example: Decrypt byte 1

If plaintext byte $\mu = 0\times5C$: RC4 & Induced

$\mu = 0\times5C$ 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 (probablistically) predict \((A', B')\)
- Ordinary likelihood calculation over only \((A', B')\)

Likelihood estimate: 

\[
\lambda_{\hat{\mu}} = (1 - \alpha(g))|w' - \hat{\mu}| \cdot \alpha(g)|\hat{\mu}|
\]
Our Goal: further kill RC4

New Biases

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

Plaintext Recovery

Break WPA-TKIP

Attack HTTPS
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
- Sender MAC
- $IV$

$IV_0, IV_1$ → Anti-FMS → Avoid weak keys which broke WEP

→ $IV$-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

Generate identical packets (otherwise MIC changes):

- Assume victim connects to server of attacker
- Retransmit identical TCP packet
  - List of plaintext candidates (unknown MIC and CRC)
  - Prune bad candidates based on CRC
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!
Our Goal: further kill RC4

New Biases

Break WPA-TKIP

Plaintext Recovery

Attack HTTPS

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

Focus on record protocol with RC4 as cipher
Targeting HTTPS Cookies

Previous attacks only used Fluhrer-McGrew (FM) biases

We combine FM bias with the ABSAB bias

Must surround cookie with known plaintext

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=aaaaaaaaaaaaaaaaaa

Headers are predictable
Surrounded by known plaintext at both sides
Preparation: manipulating cookies

a.site.com  Client  fake.site.com

HTTPS

Remove & inject *secure* cookies!
Performing the attack!

**JavaScript: Cross-Origin requests in WebWorkers**

**STEP 1**
Attacker injects code to generate requests
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
Decrypting 16-character cookie

$2^{23}$ candidates
1 candidate

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