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

Mathy Vanhoef and Frank Piessens, KU Leuven

USENIX Security 2015

DistriNet

iMinds KU LEUVEN

RC4

Intriguingly simple stream cipher

WEP

WPA-TKIP

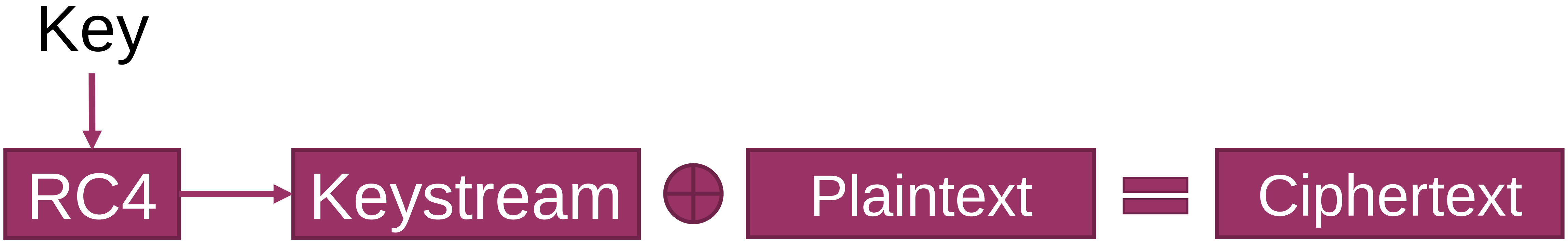
SSL / TLS

PPP/MPPE

And others ...

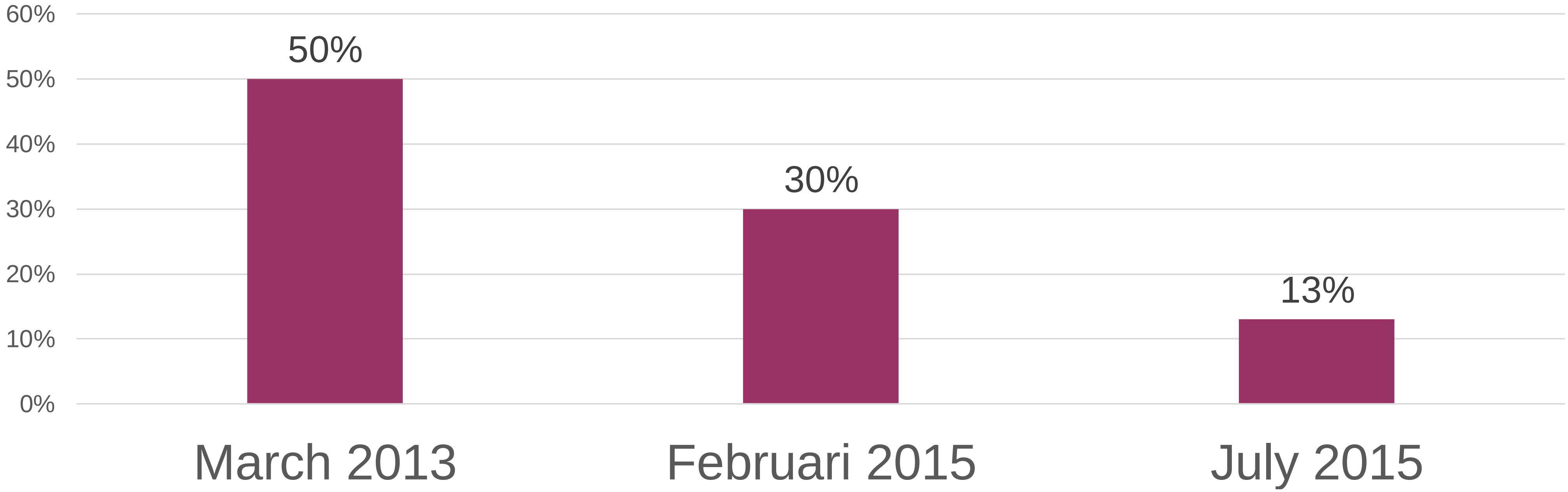
RC4

Intriguingly simple stream cipher



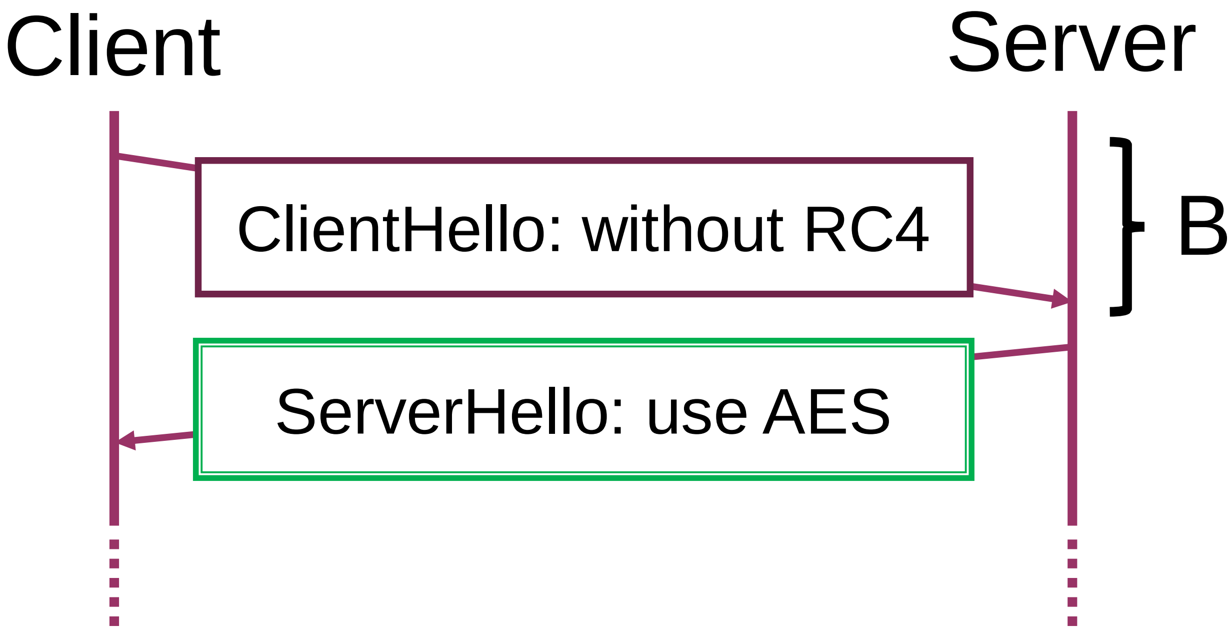
Is RC4 still used?!

ICSI Notary: TLS connections using RC4



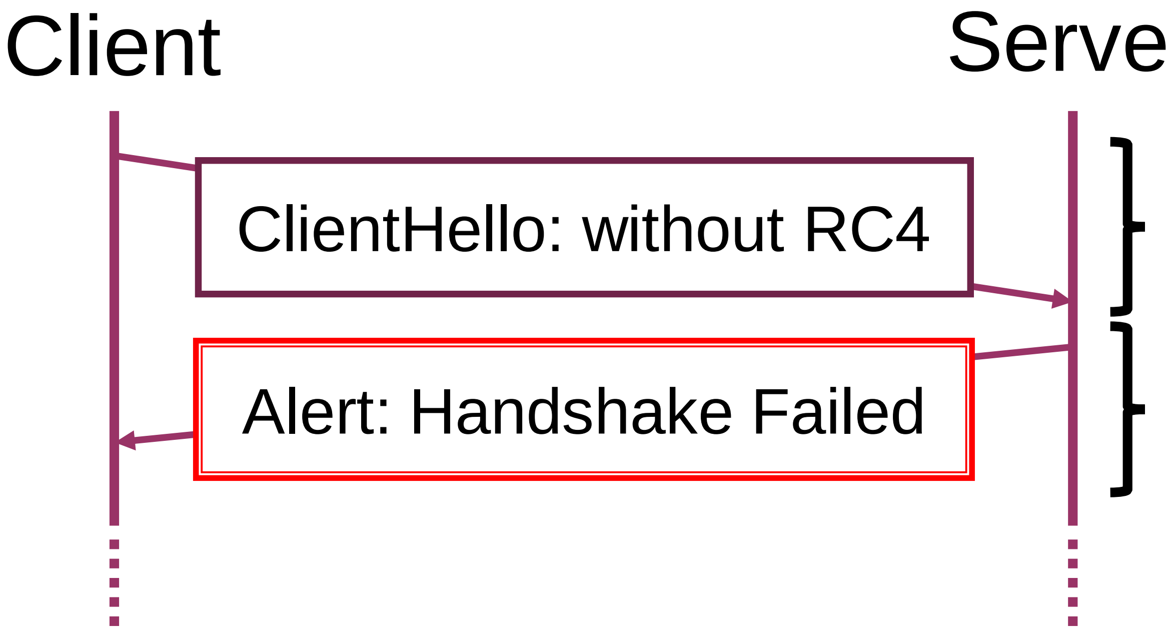
RC4 fallback not taken into account!

RC4 Fallback



Browser first tries without RC4

RC4 Fallback



Client

Server

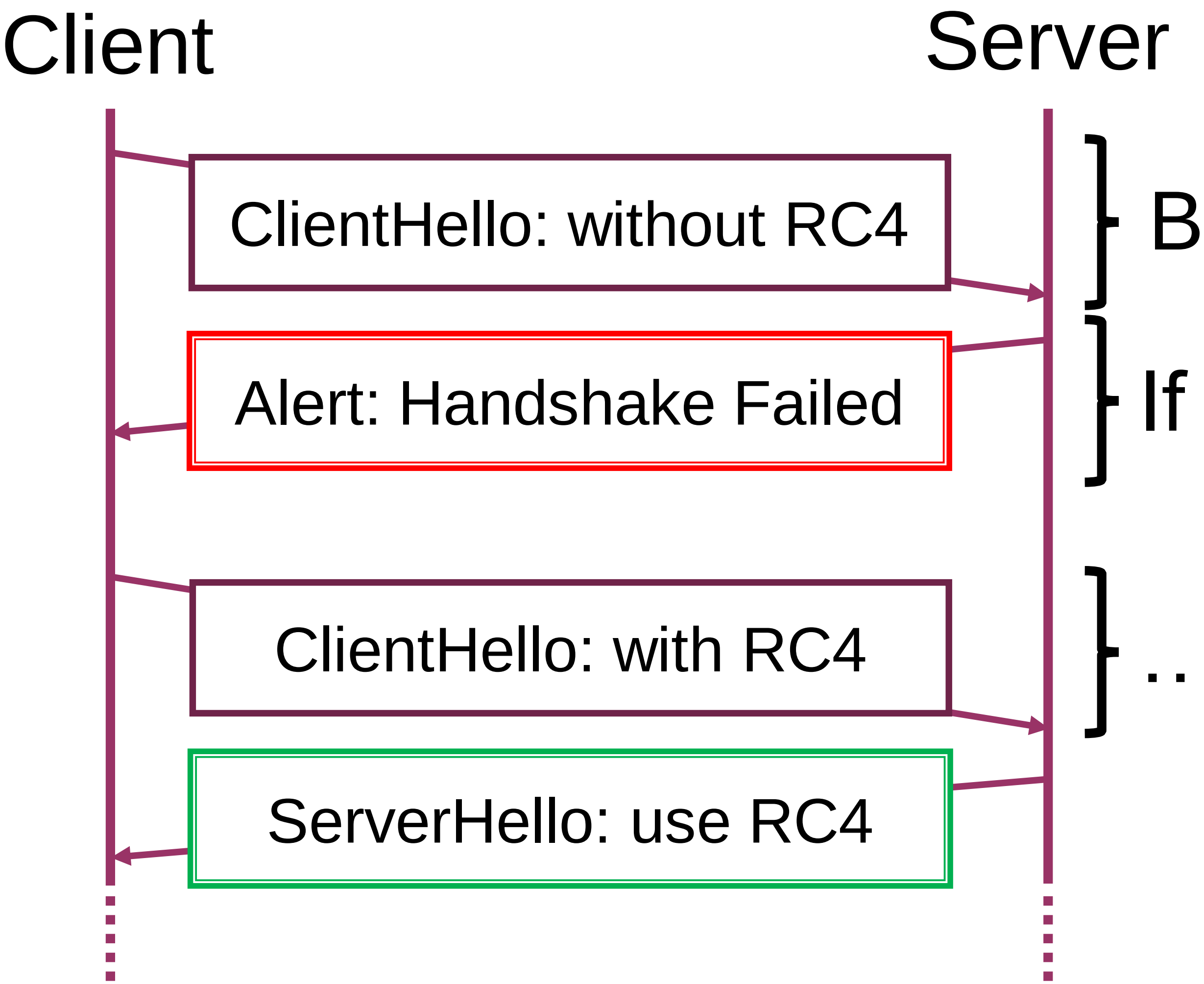
ClientHello: without RC4

Alert: Handshake Failed

Browser first tries without RC4

If that fails …

RC4 Fallback

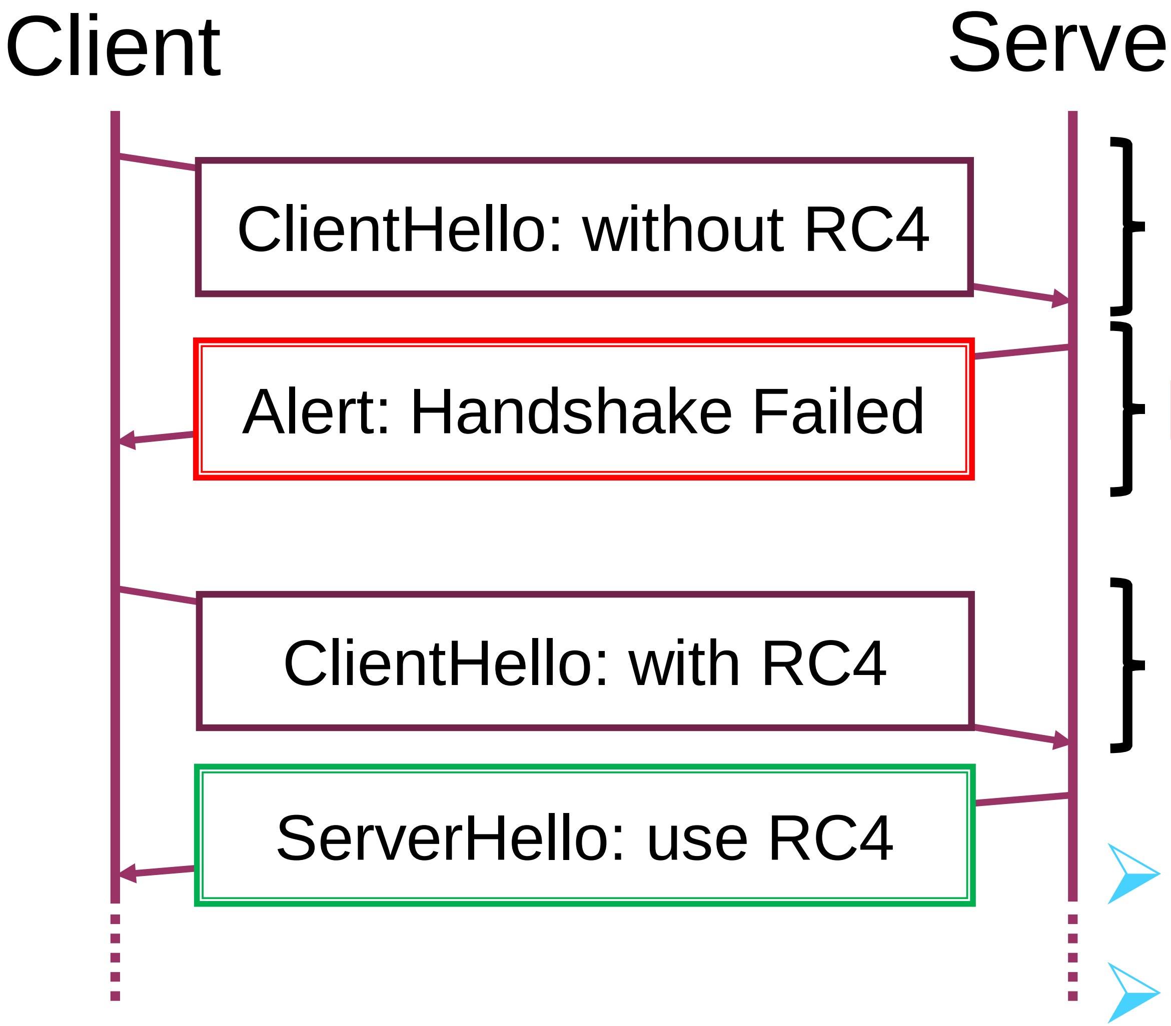


Browser first tries without RC4

If that fails …

… fallback to RC4

RC4 Fallback



Client

Server

ClientHello: without RC4

Alert: Handshake Failed

ClientHello: with RC4

ServerHello: use RC4

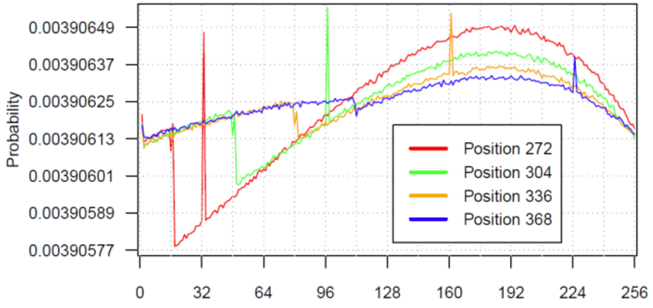
Browser first tries without RC4

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

$$\lambda_{\widehat{\mu}} = (1-\alpha(g))^{|C|-|\widehat{u}|} \cdot \alpha(g)^{|\widehat{\mu}|}$$

Plaintext Recovery

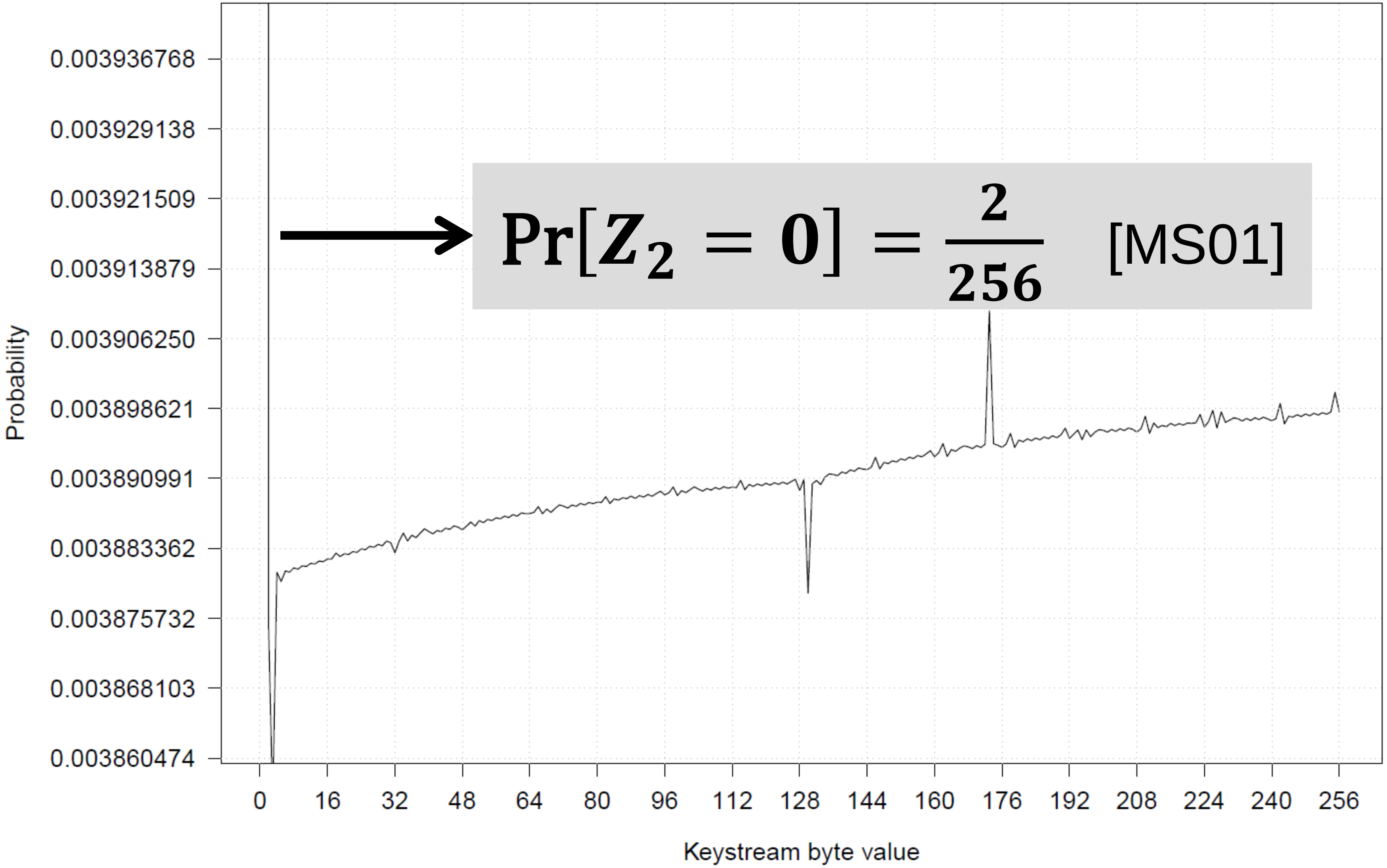
Break WPA-TKIP

Attack HTTPS

DistriNet iMinds KU LEUVEN

First: Existing Biases

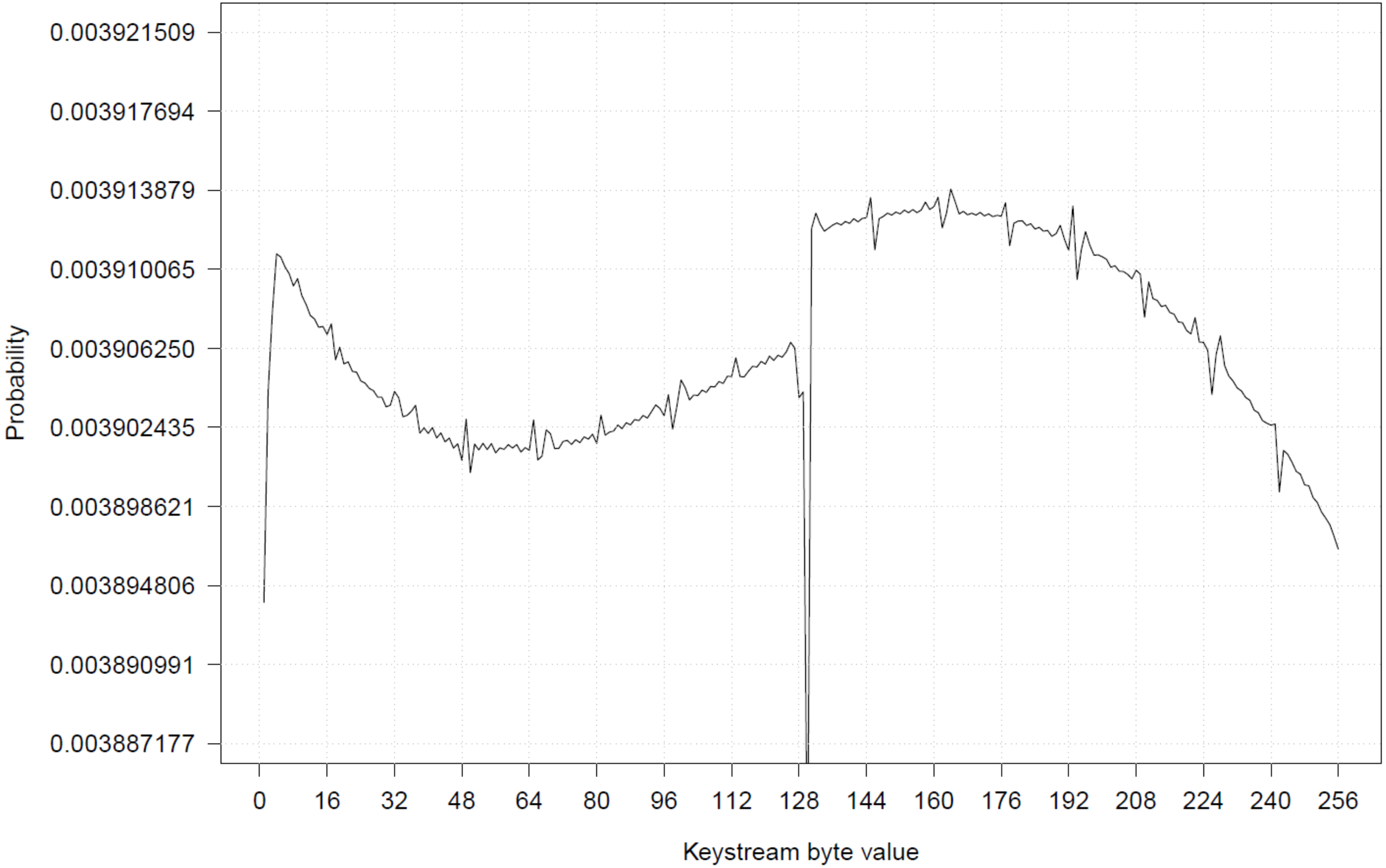
Distribution keystream byte 2



$$\Pr[Z_2 = 0] = \frac{2}{256}$$ [MS01]

First: Existing Biases

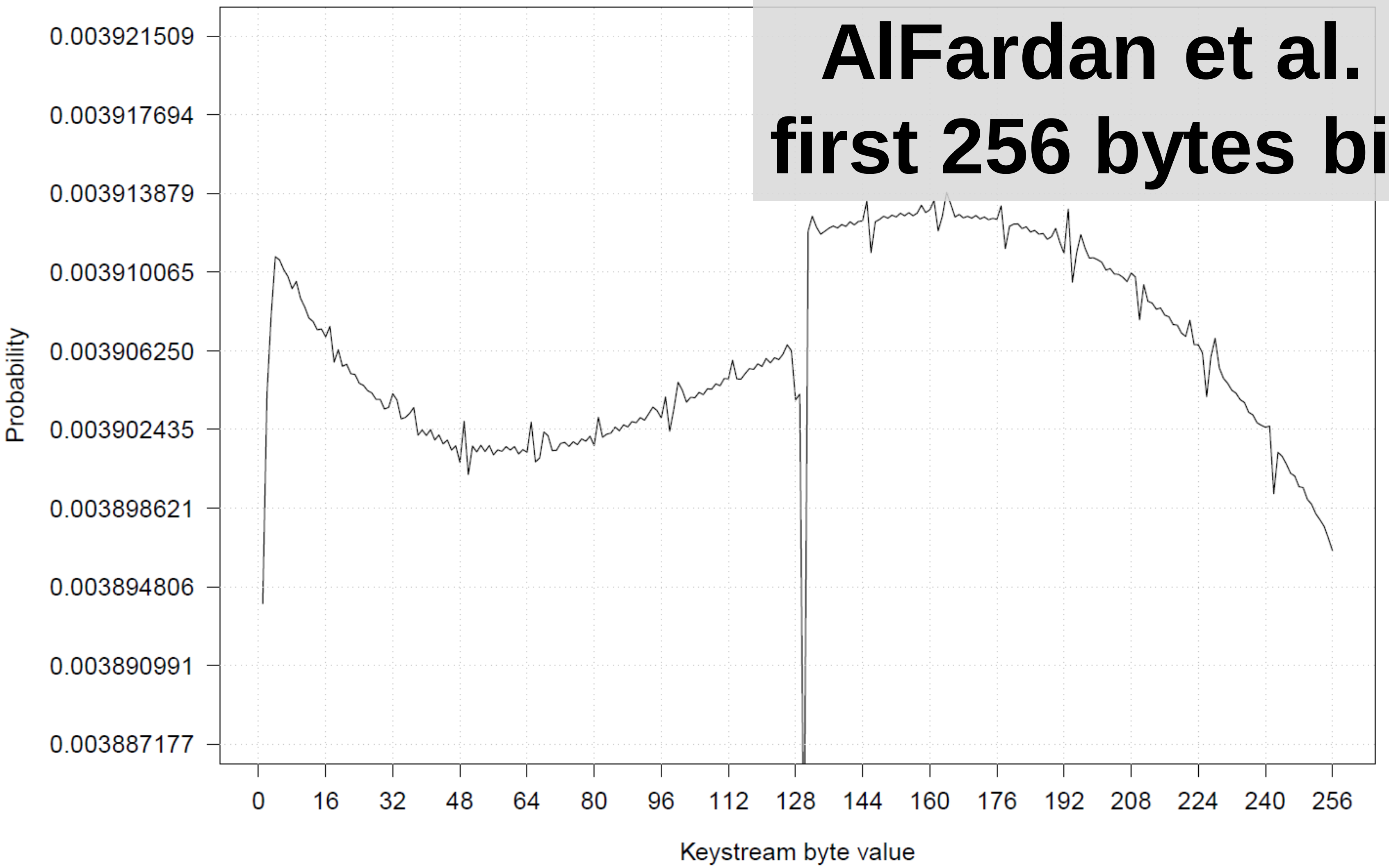
Distribution keystream byte 1



First: Existing Biases

Short-term biases

Distribution keystream byte 1 (to 256)



AlFardan et al. ‘13: first 256 bytes biased

Long-Term Biases

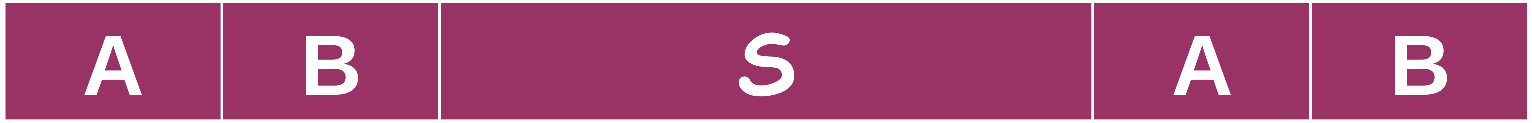
Fluhrer-McGrew (2000):

- Some consecutive values are biased

Examples: $(0, 0)$ and $(0, 1)$

Mantin's AB*S*AB Bias (2005):

- A byte pair (A, B) likely reappears



A	B	S	A	B

Search for new biases

Traditional emperical approach:

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

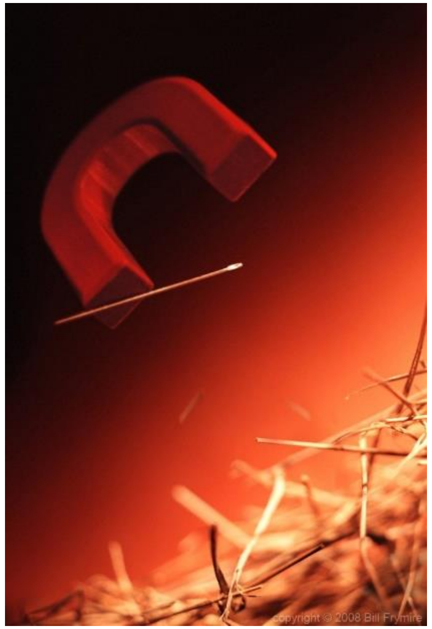
Fluhrer-McGrew: only 8 out of 65 536 pairs are biased

How to automate the search?

Search for new biases

Traditional emperical approach:

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



copyright © 2008 Bill Frymire

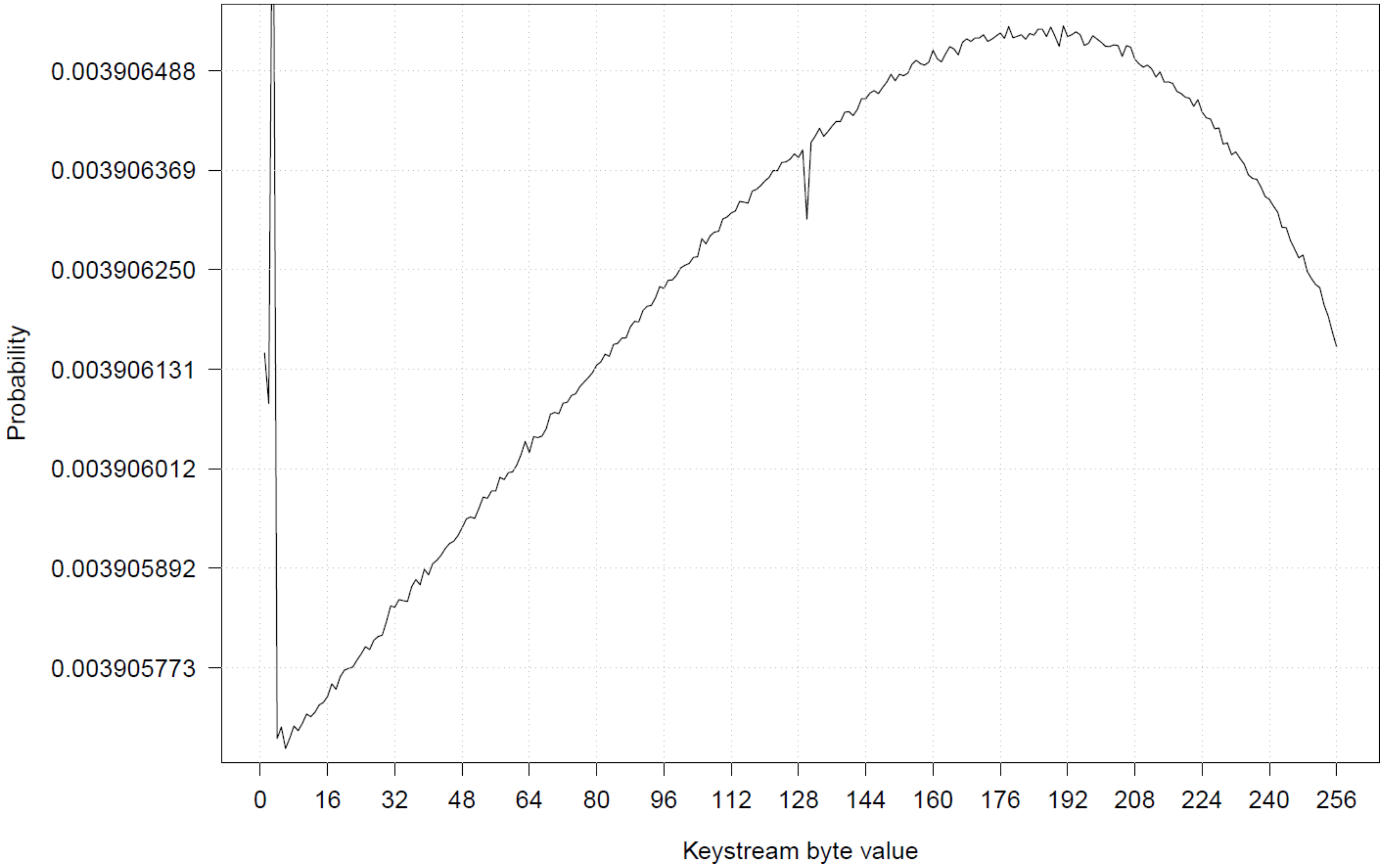
Hypothesis tests!

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

→ Allows a large-scale search,
revealing many new biases

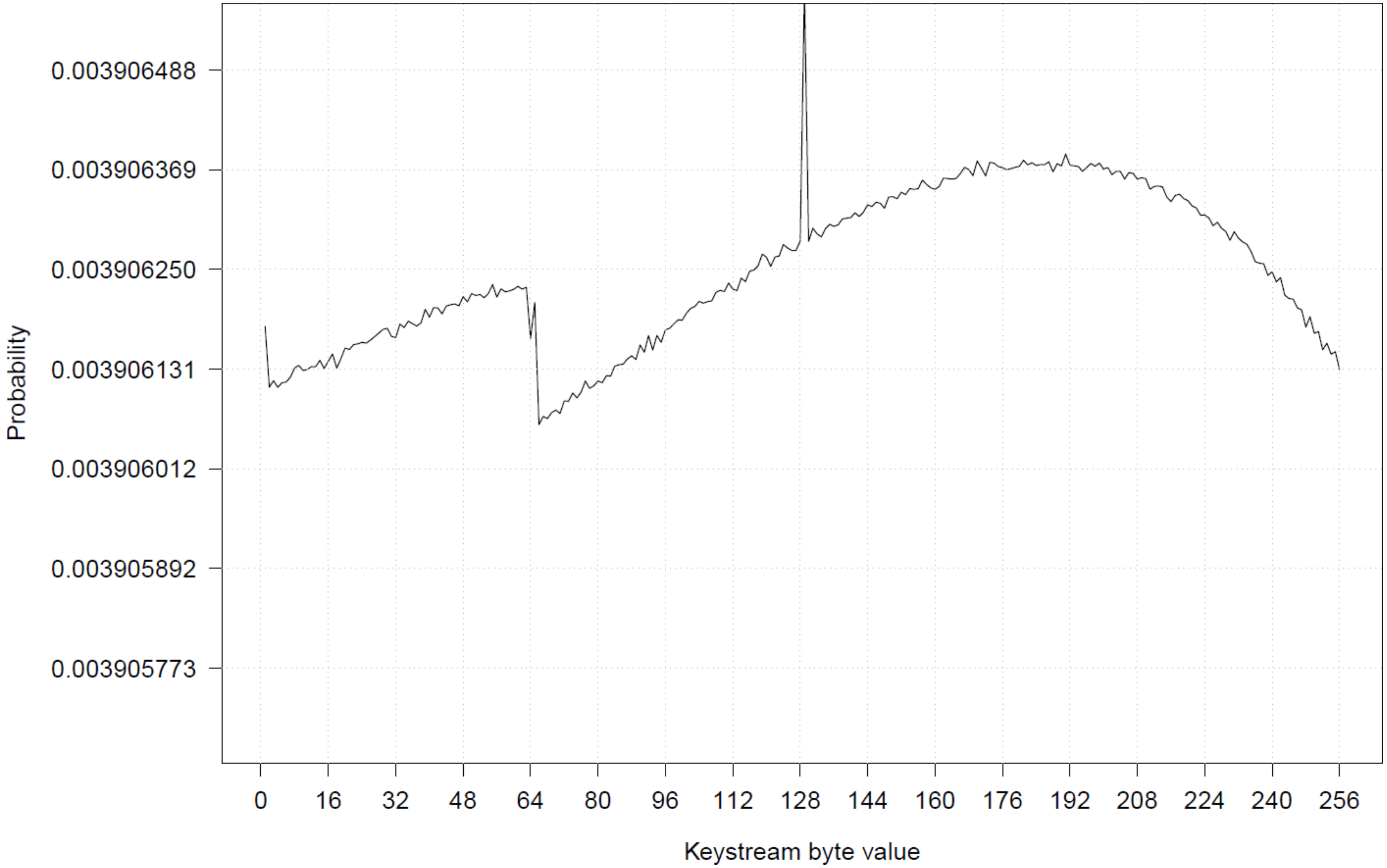
Biases in Bytes 258-513

Example: keystream byte 258



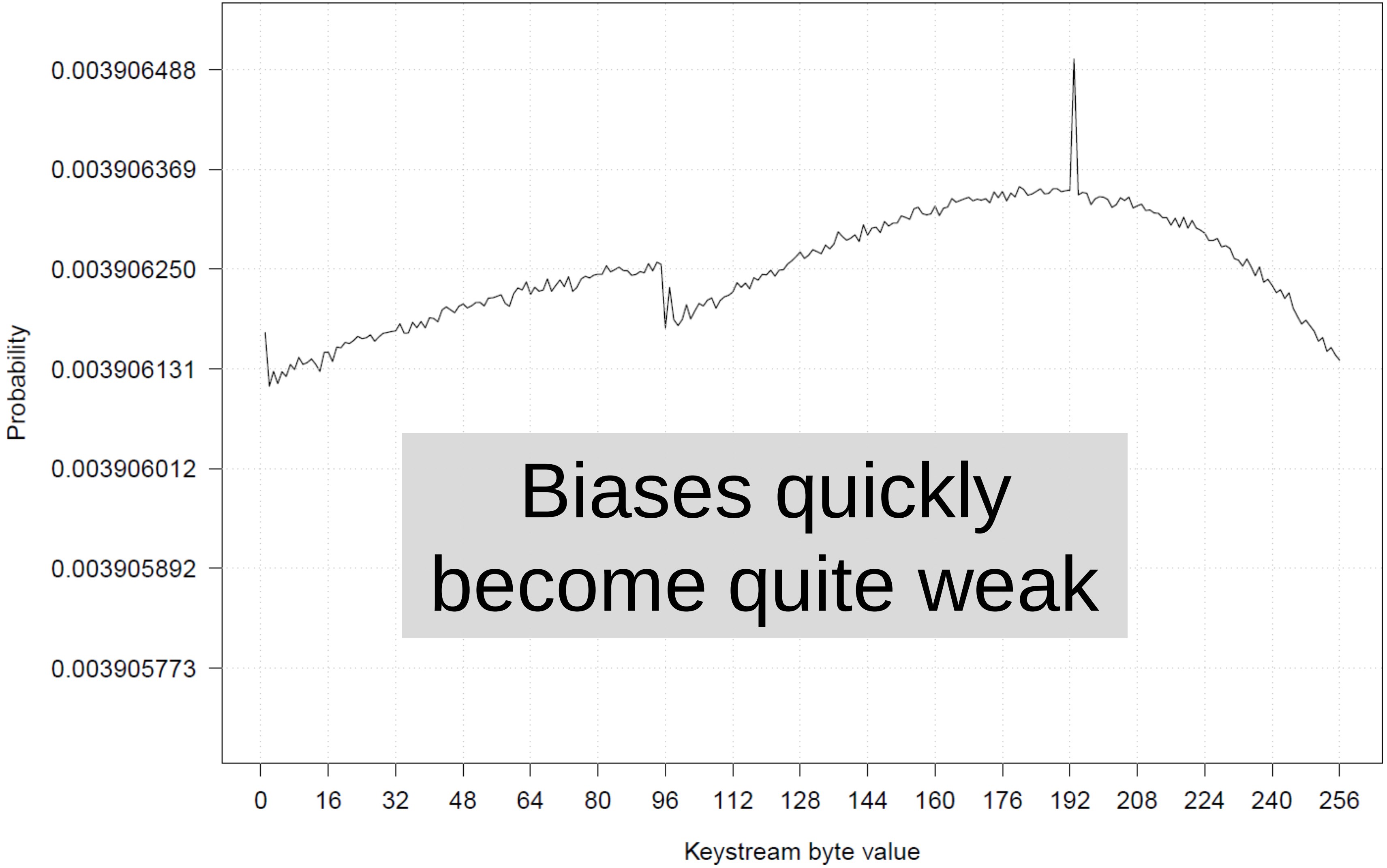
Biases in Bytes 258-513

Example: keystream byte 320



Biases in Bytes 258-513

Example: keystream byte 352

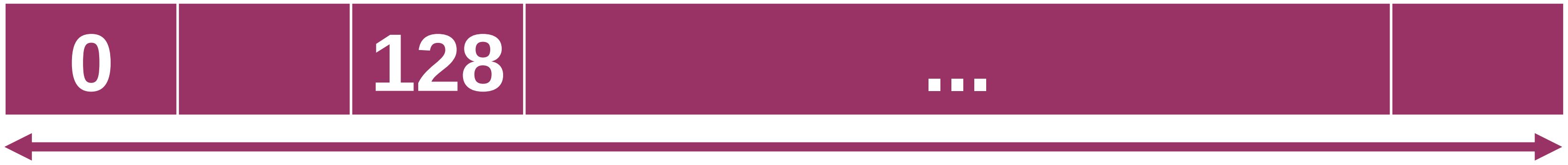


Biases quickly become quite weak

New Long-term Bias

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

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

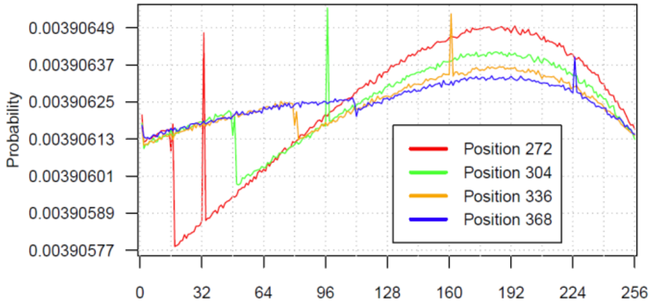


Every block of 256 bytes

Additional Biases

See paper!

Our Goal: further kill RC4



New Biases

$$\lambda_{\widehat{\mu}} = (1-\alpha(g))^{|\mathcal{C}|-|\widehat{u}|} \cdot \alpha(g)^{|\widehat{\mu}|}$$

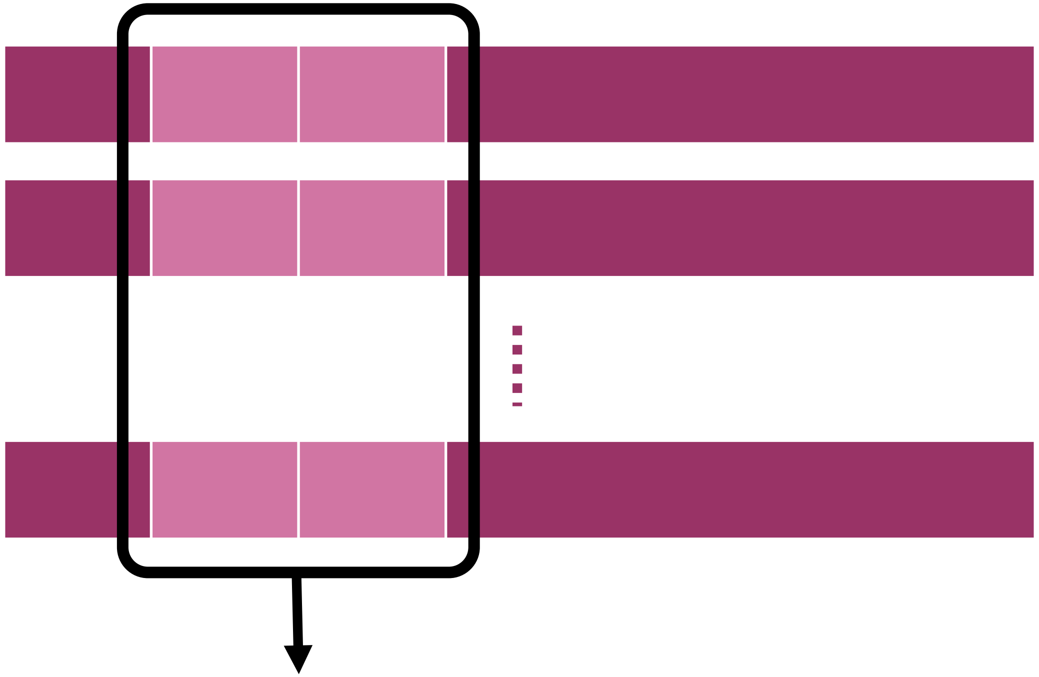
Plaintext Recovery

Break WPA-TKIP

Attack HTTPS

Existing Methods [AlFardan et al. ‘13]

Plaintext encrypted under several keystreams

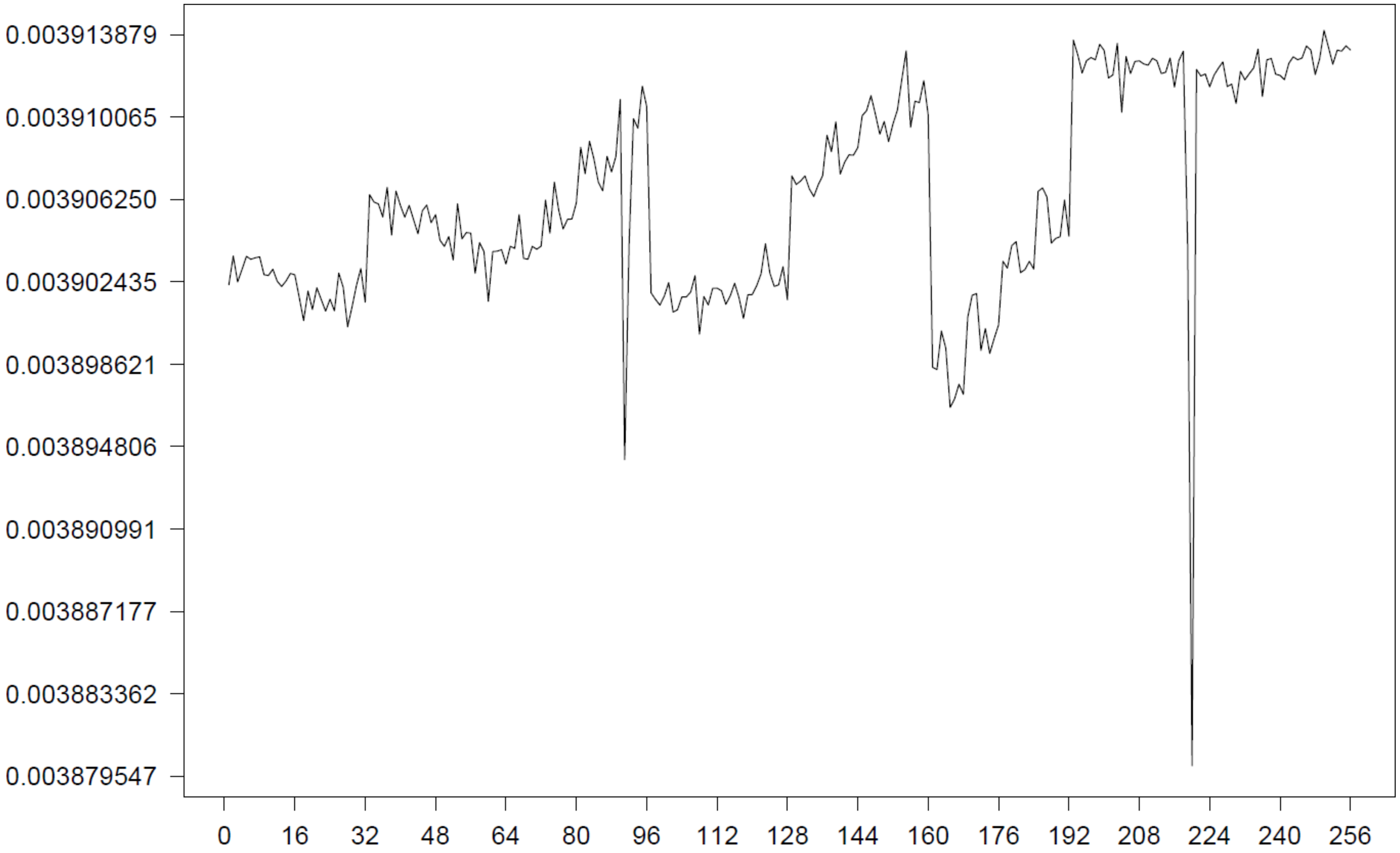


Verify guess: how close to real keystream distribution?

Ciphertext Distribution ⊕ Plaintext guess μ = **Induced** keystream distribution

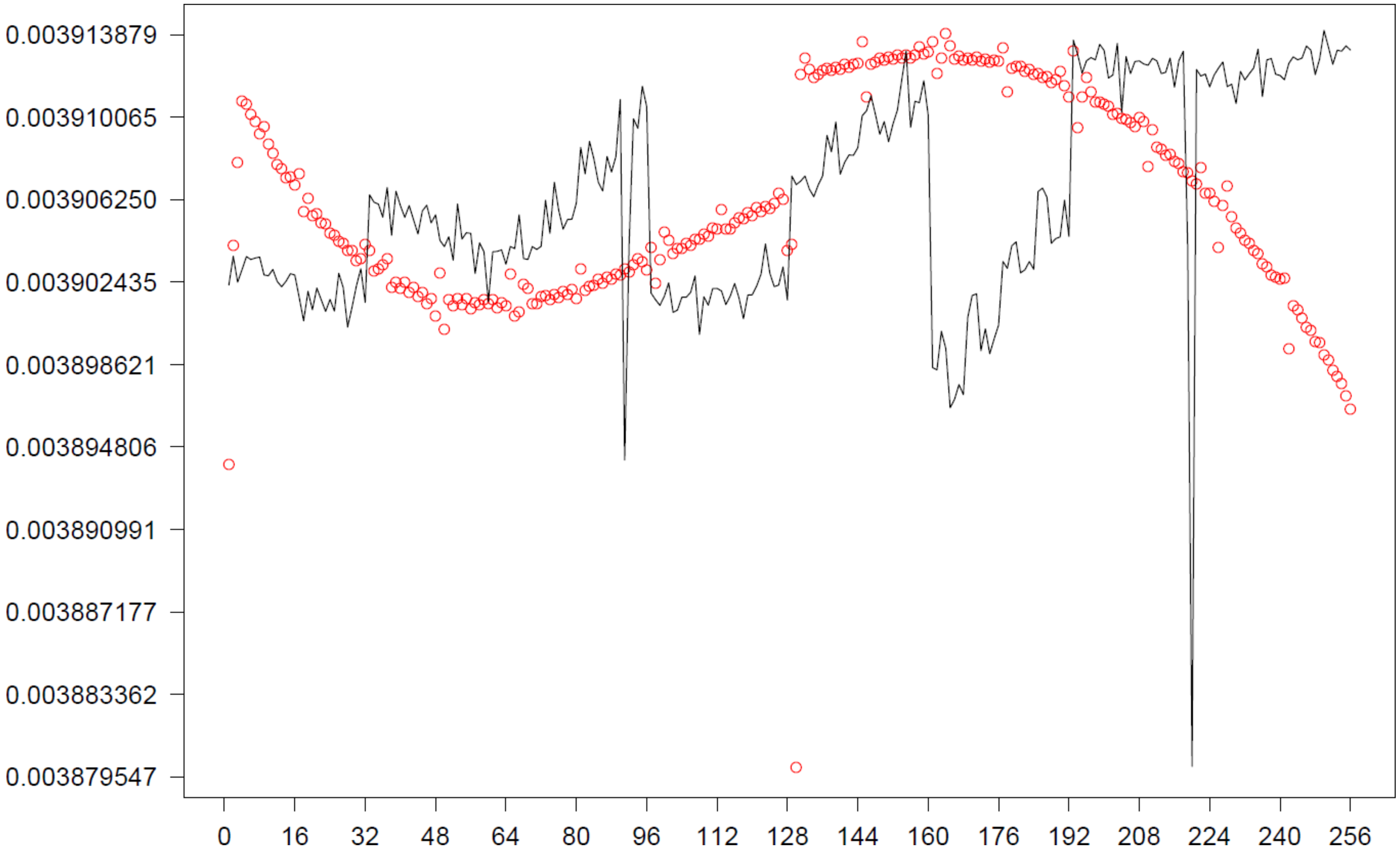
Example: Decrypt byte 1

Ciphertext Distribution



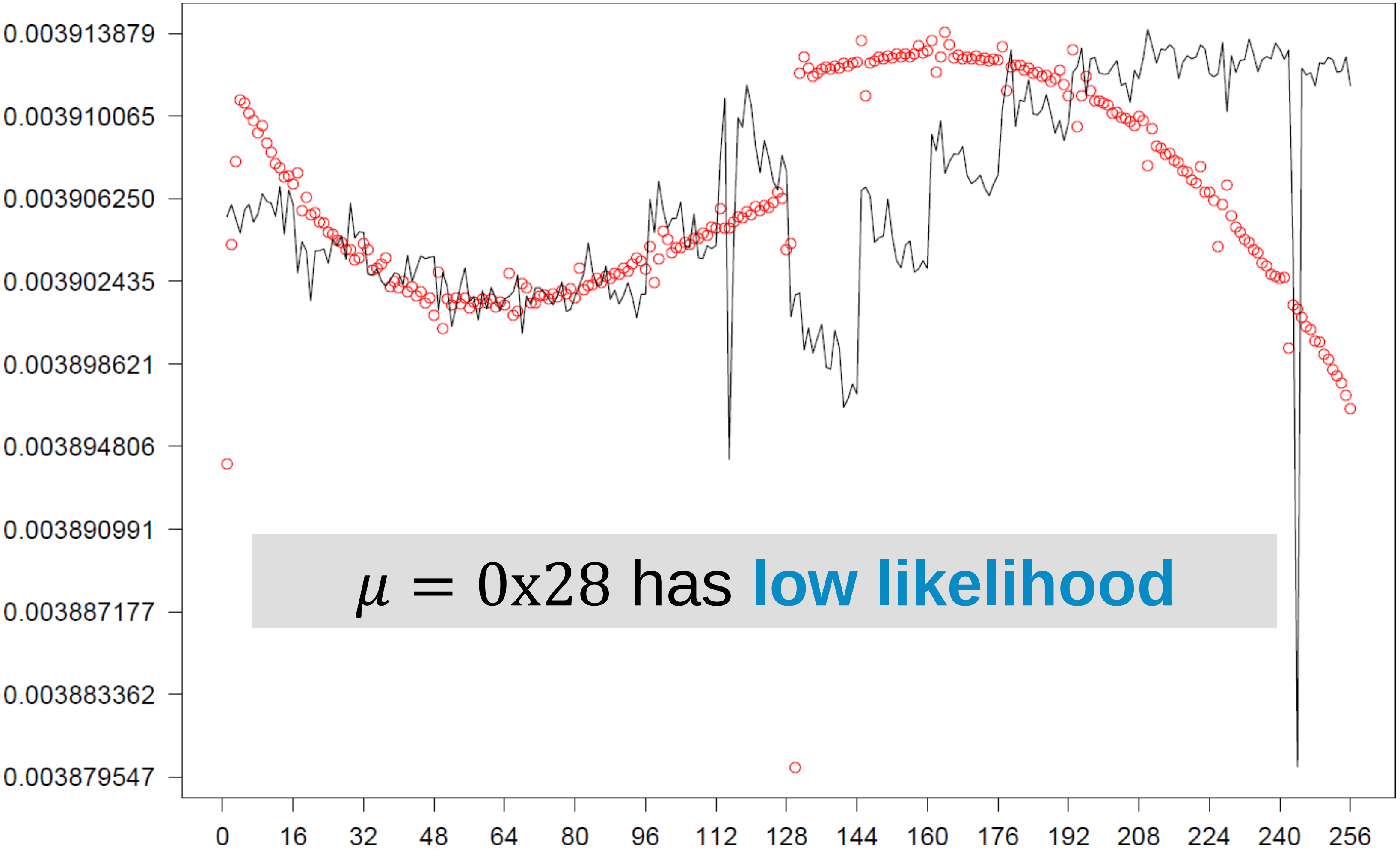
Example: Decrypt byte 1

RC4 & Ciphertext distribution



Example: Decrypt byte 1

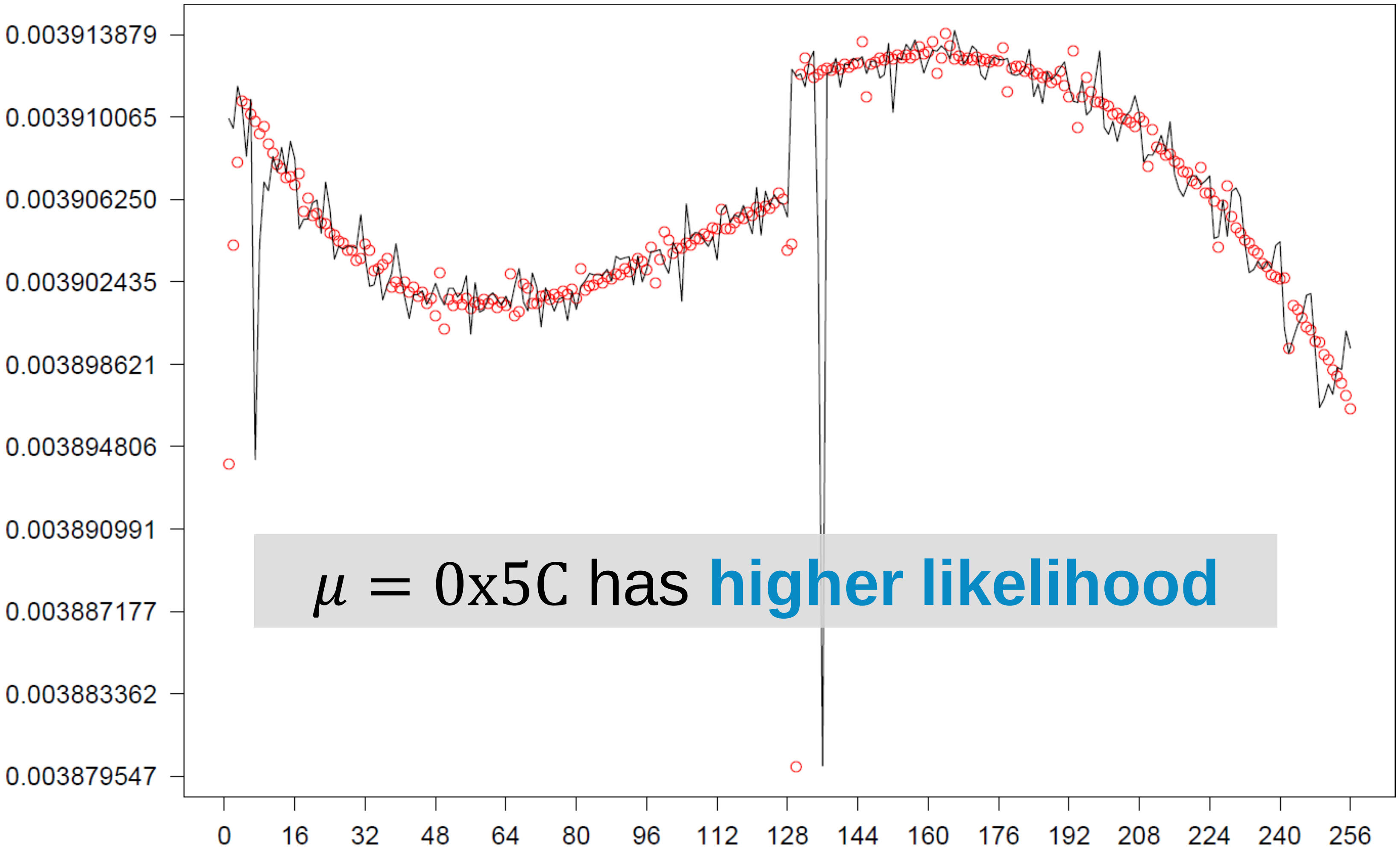
If plaintext byte $\mu = 0\text{x}28$: **RC4** & **Induced**



$\mu = 0\text{x}28$ has **low likelihood**

Example: Decrypt byte 1

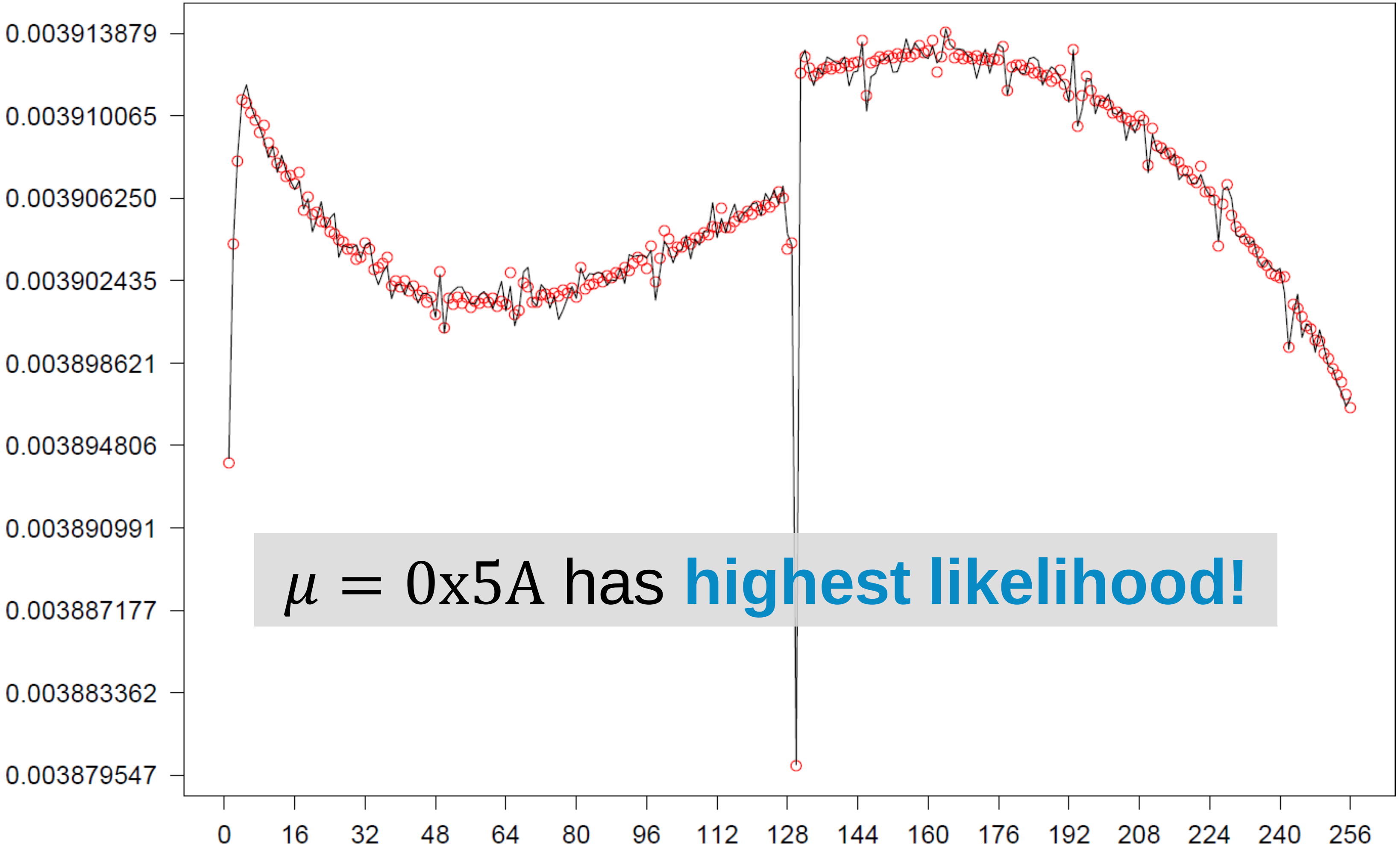
If plaintext byte $\mu = \mathrm{0x5C}$: **RC4** & **Induced**



$\mu = \mathrm{0x5C}$ has **higher likelihood**

Example: Decrypt byte 1

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



$\mu = \mathrm{0x5A}$ has **highest likelihood!**

Types of likelihood estimates

Previous works: pick value with highest likelihood.

Better idea: list of candidates in decreasing likelihood:

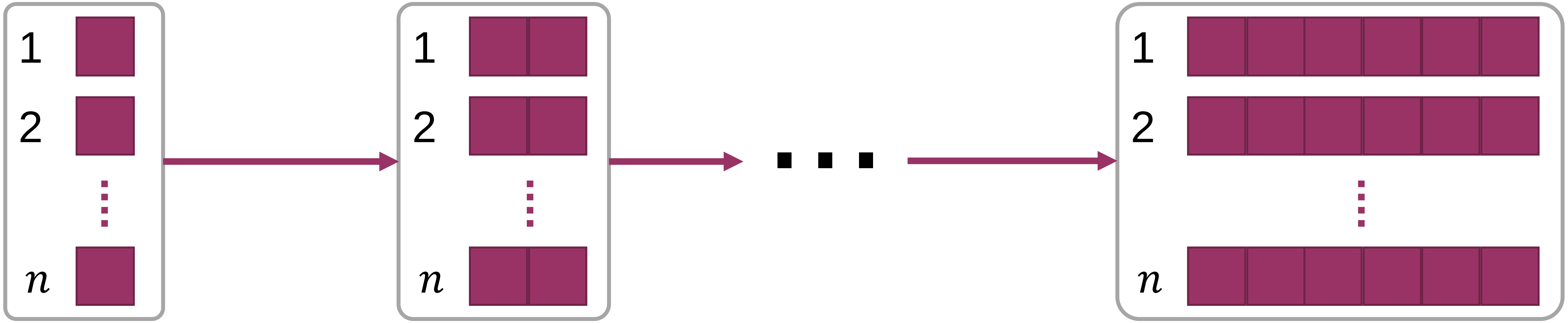
- Most likely one may not be correct!
- Prune bad candidates (e.g. bad CRC)
- Brute force cookies or passwords

How to calculate list of candidates?

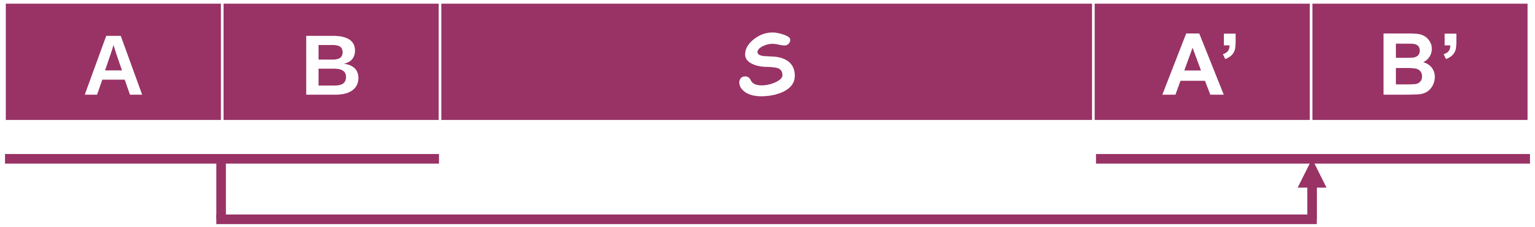
1st idea: Generate List of Candidatess

Gist of the Algorithm: Incremental approach

Calculate candidates of length 1, length 2, ...



2nd idea: abusing the ABSAB bias



A B S A’ B’

Known Plaintext Unknown Plaintext

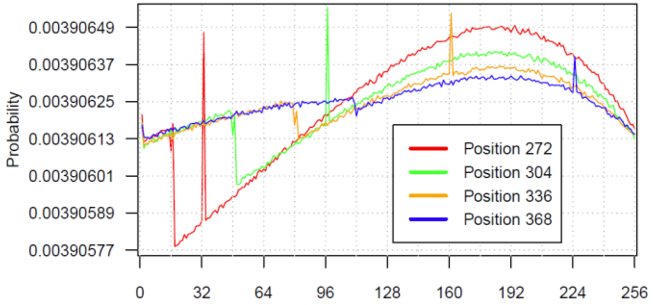
Assume there’s **surrounding known plaintext** !

- Derive values of (A, B)
- Combine with ABSAB bias to (probablisticly) predict $(\mathrm{A}', \mathrm{B}')$

➢ Ordinary likelihood calculation over only $(\mathrm{A}', \mathrm{B}')$

Likelihood estimate: $\lambda_{\widehat{\mu}} = (1-\alpha(g))^{|\mathcal{C}|-|\widehat{u}|} \cdot \alpha(g)^{|\widehat{\mu}|}$

Our Goal: further kill RC4



New Biases

$$\lambda_{\widehat{\mu}} = (1-\alpha(g))^{|\mathcal{C}|-|\widehat{u}|} \cdot \alpha(g)^{|\widehat{\mu}|}$$

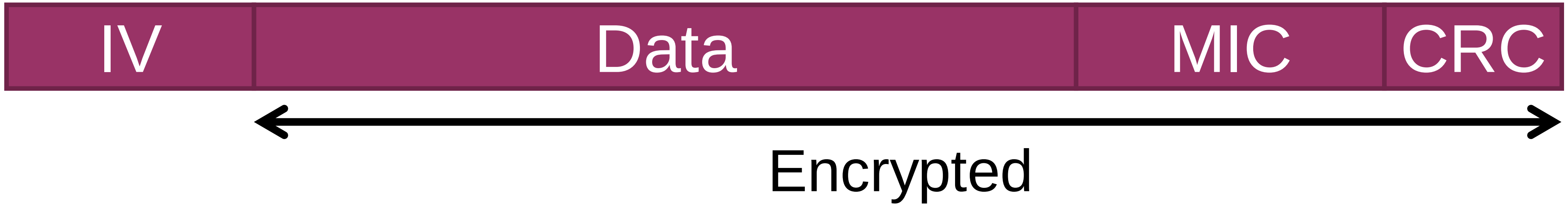
Plaintext Recovery

Break WPA-TKIP

Attack HTTPS

TKIP Background

How are packets sent/received?

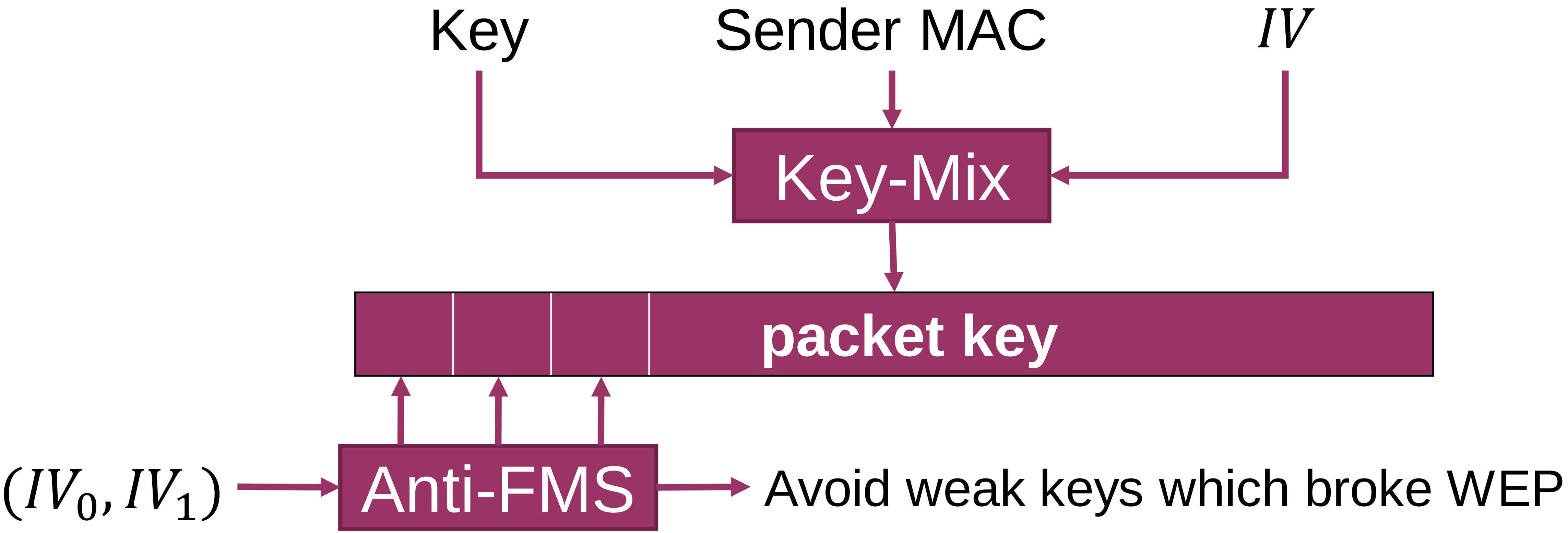


IV | Data | MIC | CRC

Encrypted

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)

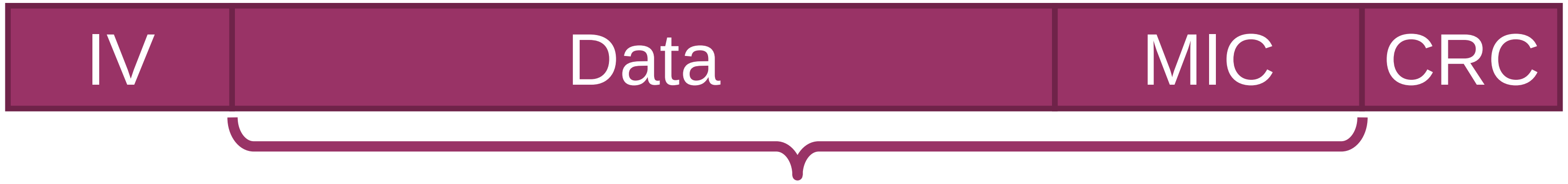
Flaw #1: TKIP Per-packet Key



→ IV-dependent biases in keystream

[Gupta/Paterson et al.]

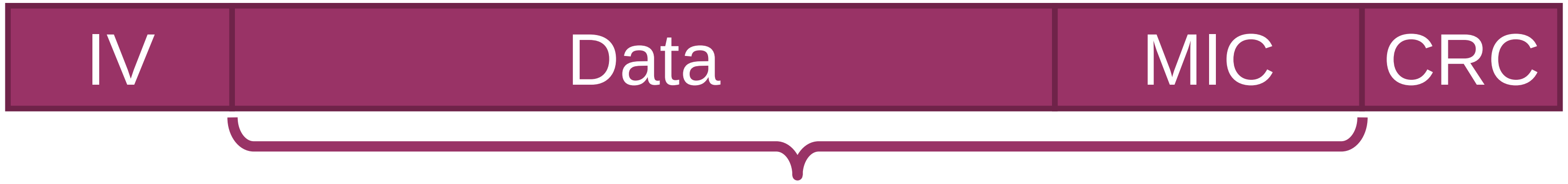
Flaw #2: MIC is invertible



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

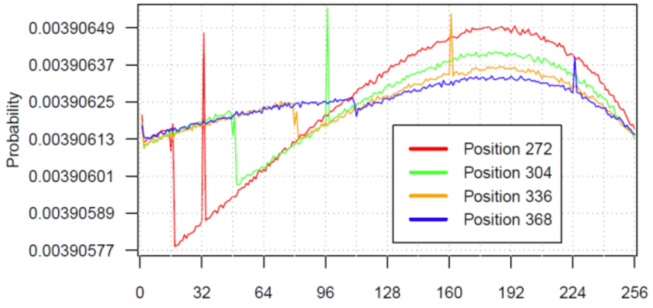
Simulations with 2^{30} candidates:

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

Emperical 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

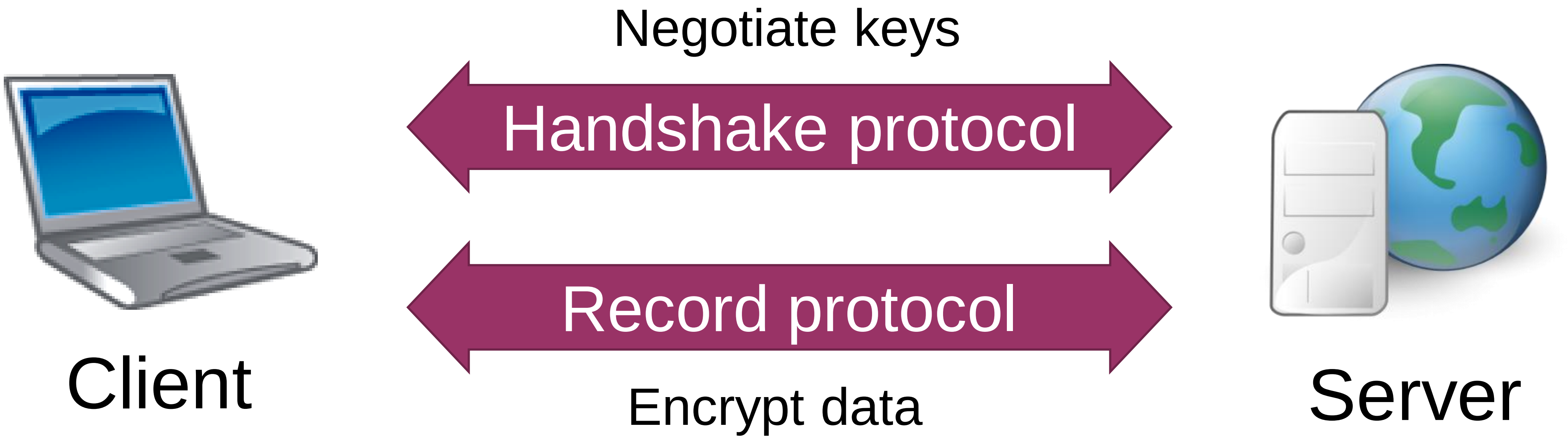
$$\lambda_{\widehat{\mu}} = (1-\alpha(g))^{|C|-|\widehat{u}|} \cdot \alpha(g)^{|\widehat{\mu}|}$$

Plaintext Recovery

Break WPA-TKIP

Attack HTTPS

TLS Background



Negotiate keys

Handshake protocol

Client

Record protocol

Encrypt data

Server

→ 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

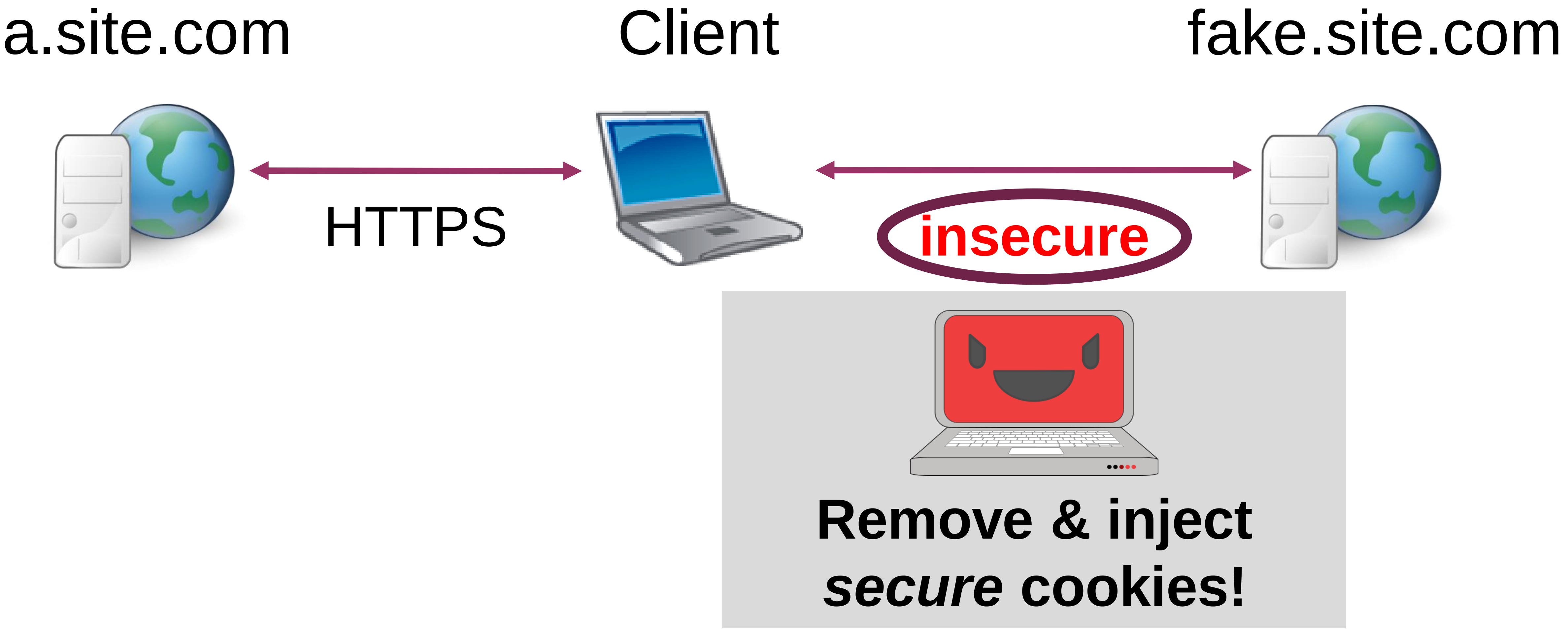
Example: manipulated HTTP request

```
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=aaaaaaaaaaaaaaaa
```

Headers are predictable

Surrounded by known plaintext at both sides

Preparation: manipulating cookies



a.site.com

Client

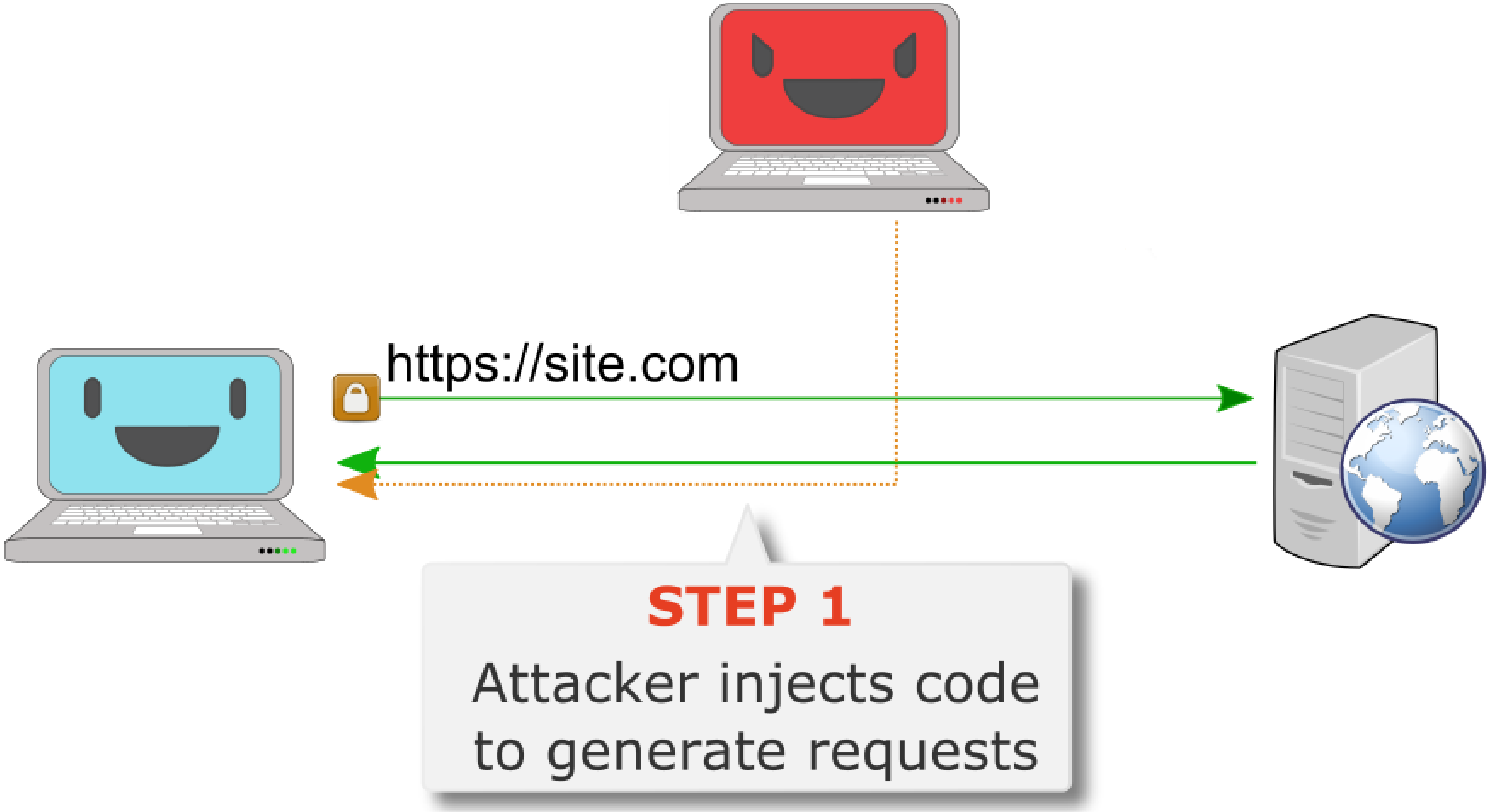
fake.site.com

HTTPS

insecure

Remove & inject
***secure* cookies!**

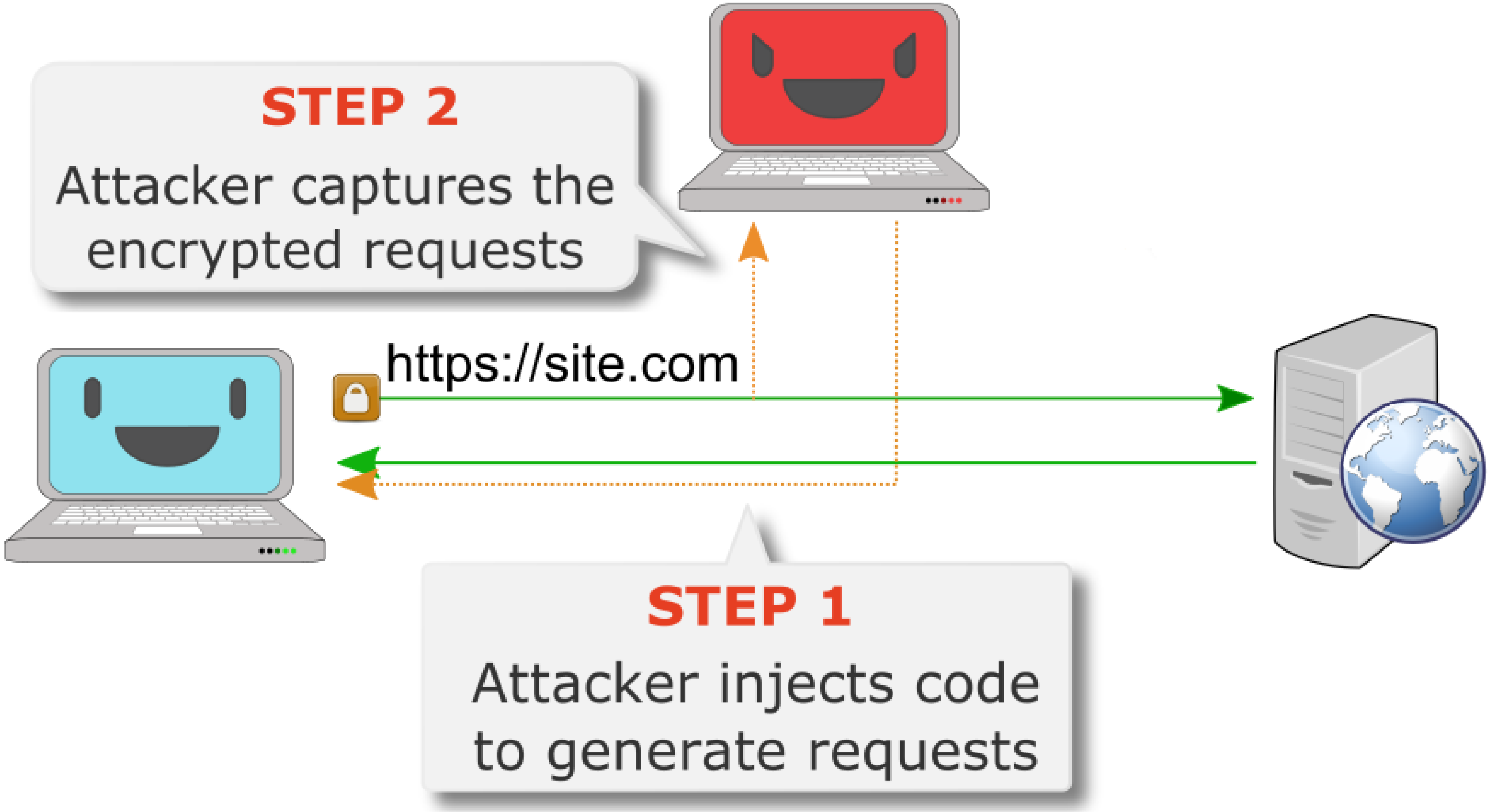
Performing the attack!



STEP 1

Attacker injects code to generate requests

Performing the attack!



STEP 2

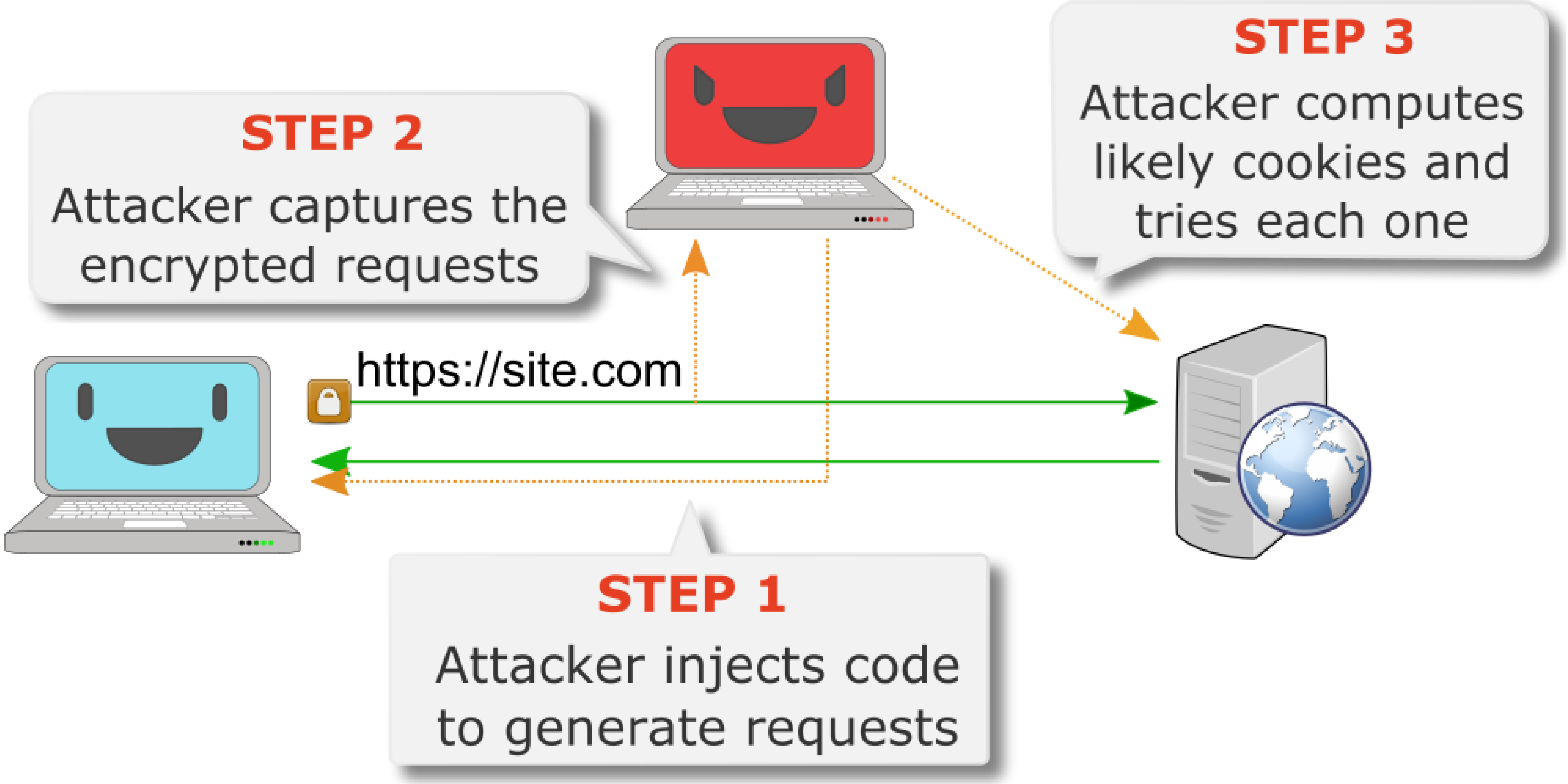
Attacker captures the encrypted requests

STEP 1

Attacker injects code to generate requests

Keep-Alive connection to generate them fast

Performing the attack!



STEP 2

Attacker captures the encrypted requests

STEP 3

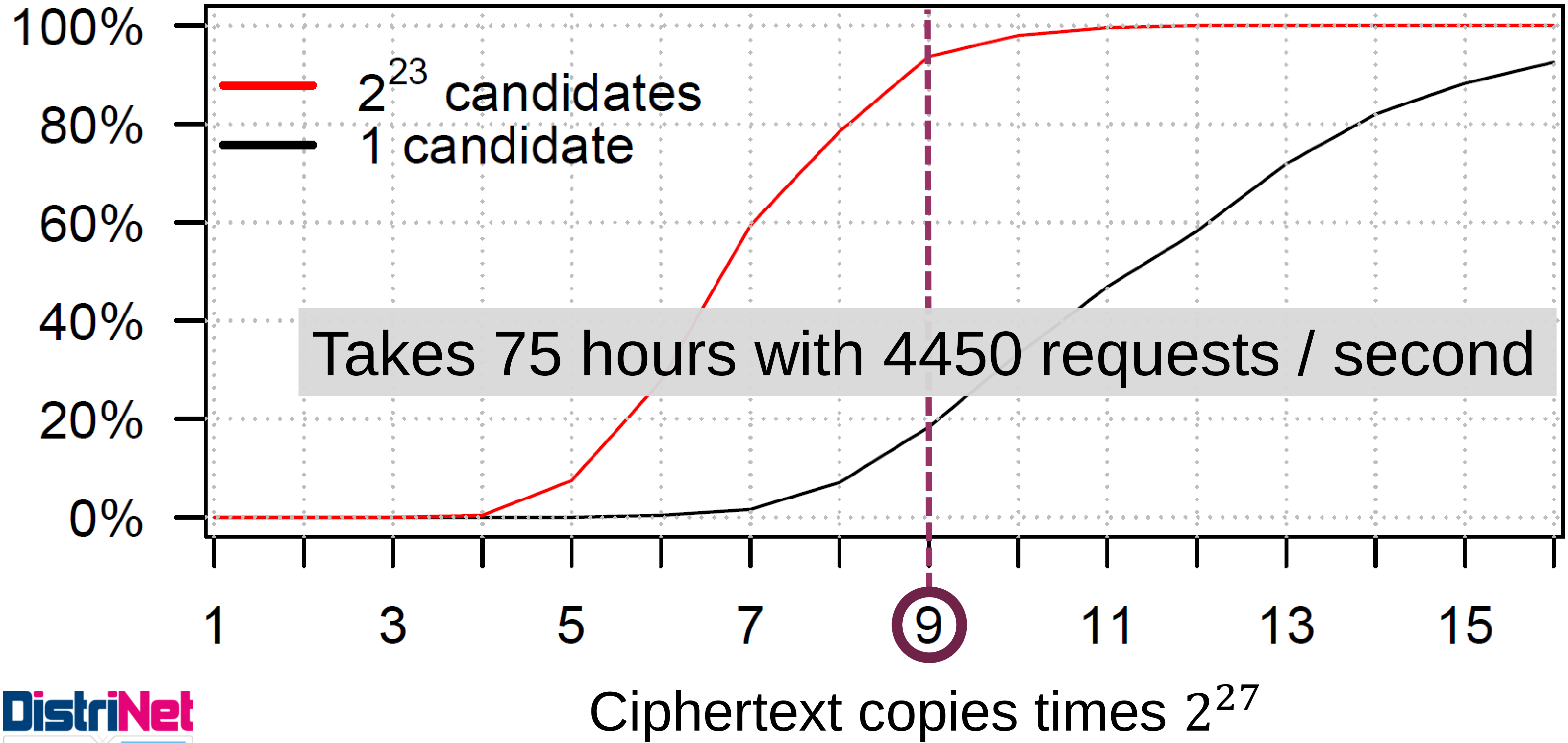
Attacker computes likely cookies and tries each one

STEP 1

Attacker injects code to generate requests

Combine Fluhrer-McGrew and ABSAB biases

Decrypting 16-character cookie



Takes 75 hours with 4450 requests / second

Decrypting 16-character cookie

DEMO!

rc4nomore.com

Questions?

May the bias be ever in your favor