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 OWASP

RC4

Intriguingly simple stream cipher

~ 10 lines in Python

WEP
WPA-TKIP

SSL / TLS

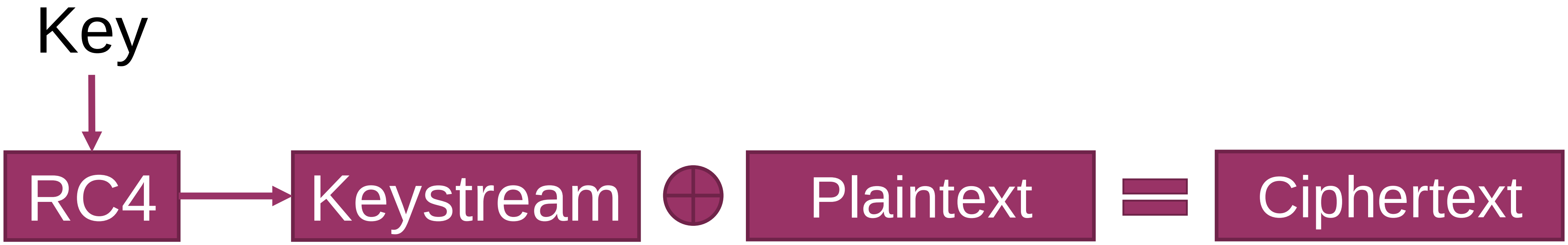
PPP/MPPE

And others ...

RC4

Intriguingly simple stream cipher

~ 10 lines in Python



High level description

Shuffles permutation of [0..255]

77	37	102	233	…	151	14	198	0	56

Secret index j

pseudo-randomly
updated value

Public index i

= keystream
position mod 256

→ Output byte selected based on index j and i

Why study RC4?

Immune to several attacks on SSL/TLS:

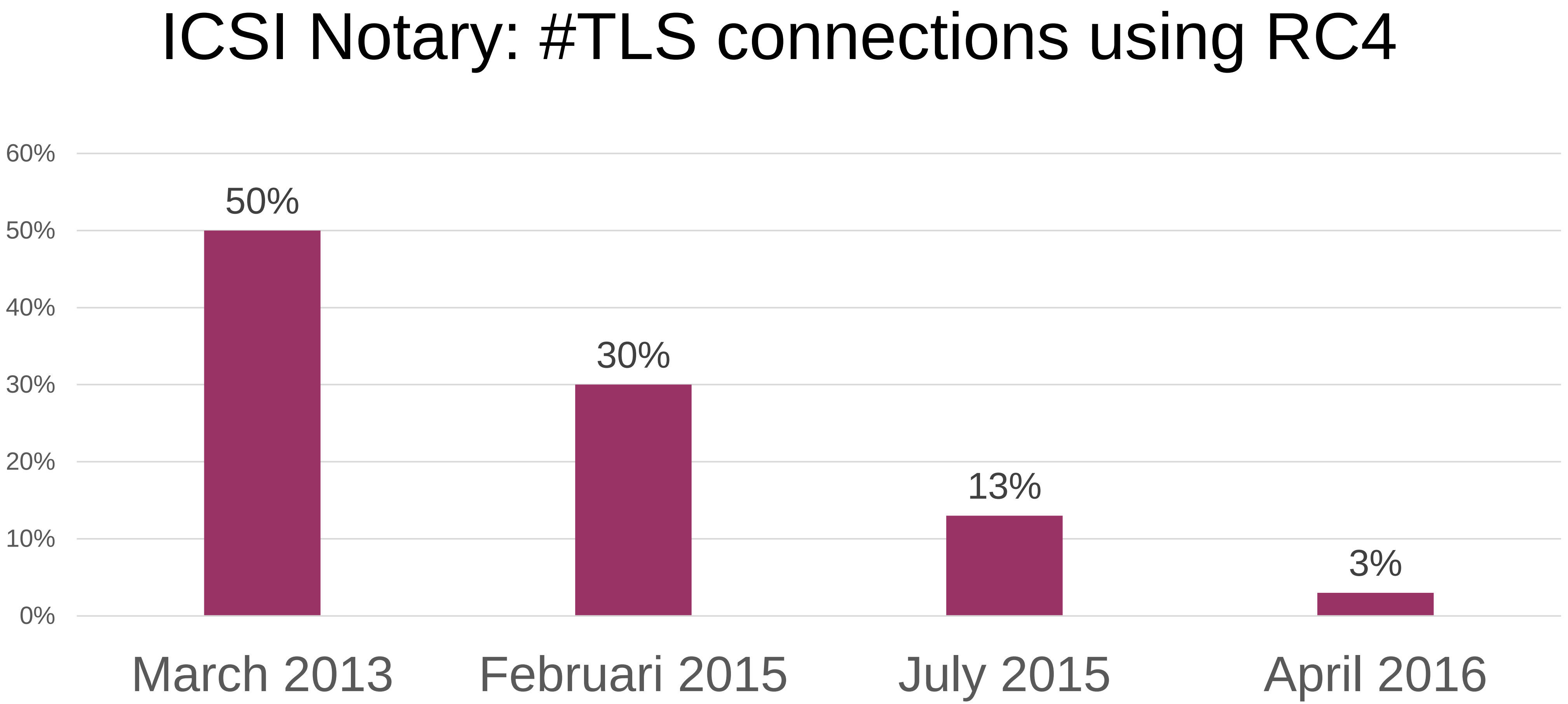
- 2003: Padding oracle
- 2011: BEAST
- 2013: Lucky 13
- 2014: POODLE

Target CBC mode encryption (block ciphers)

- Solution: use stream cipher or up-to-date TLS library
- Only widely supported option was RC4

DistriNet iMinds KU LEUVEN

RC4 was heavily used!



Browser support today (April 2016)

Chrome: dropped support in v48 (20 Jan. 2016)

Firefox: dropped support in v44 (26 Jan. 2016)

IE11: supports RC4

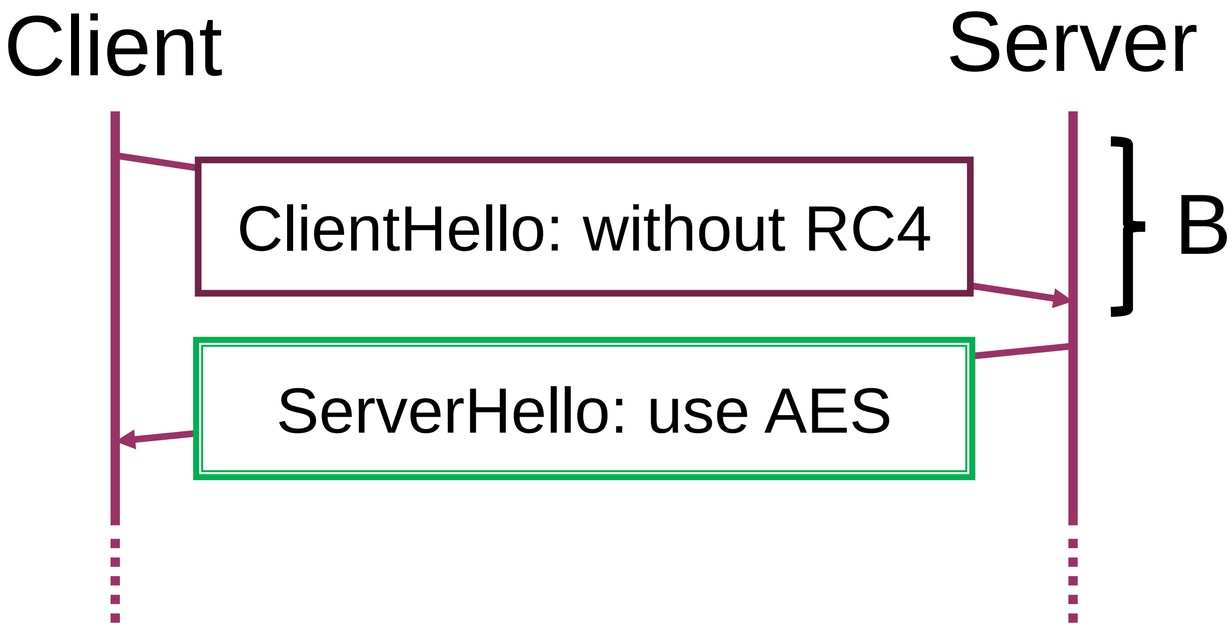
Edge: supports RC4

“will be disabled in forthcoming update”

Has fallback to RC4

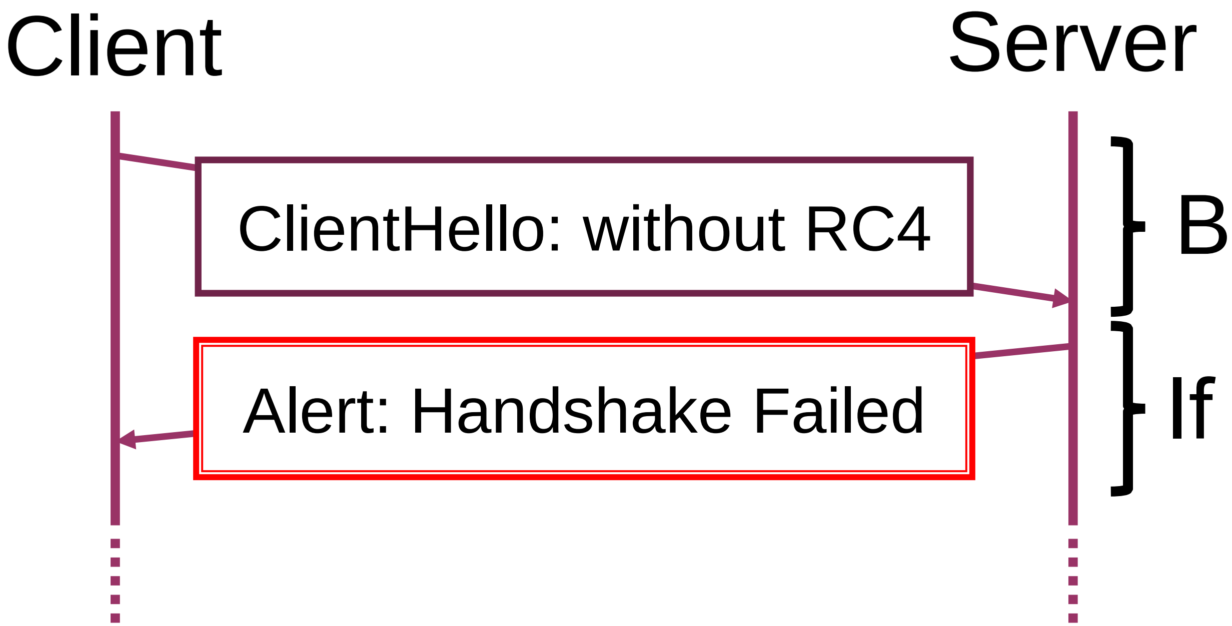
DistriNet iMinds KU LEUVEN

Fallback to RC4



Browser first tries without RC4

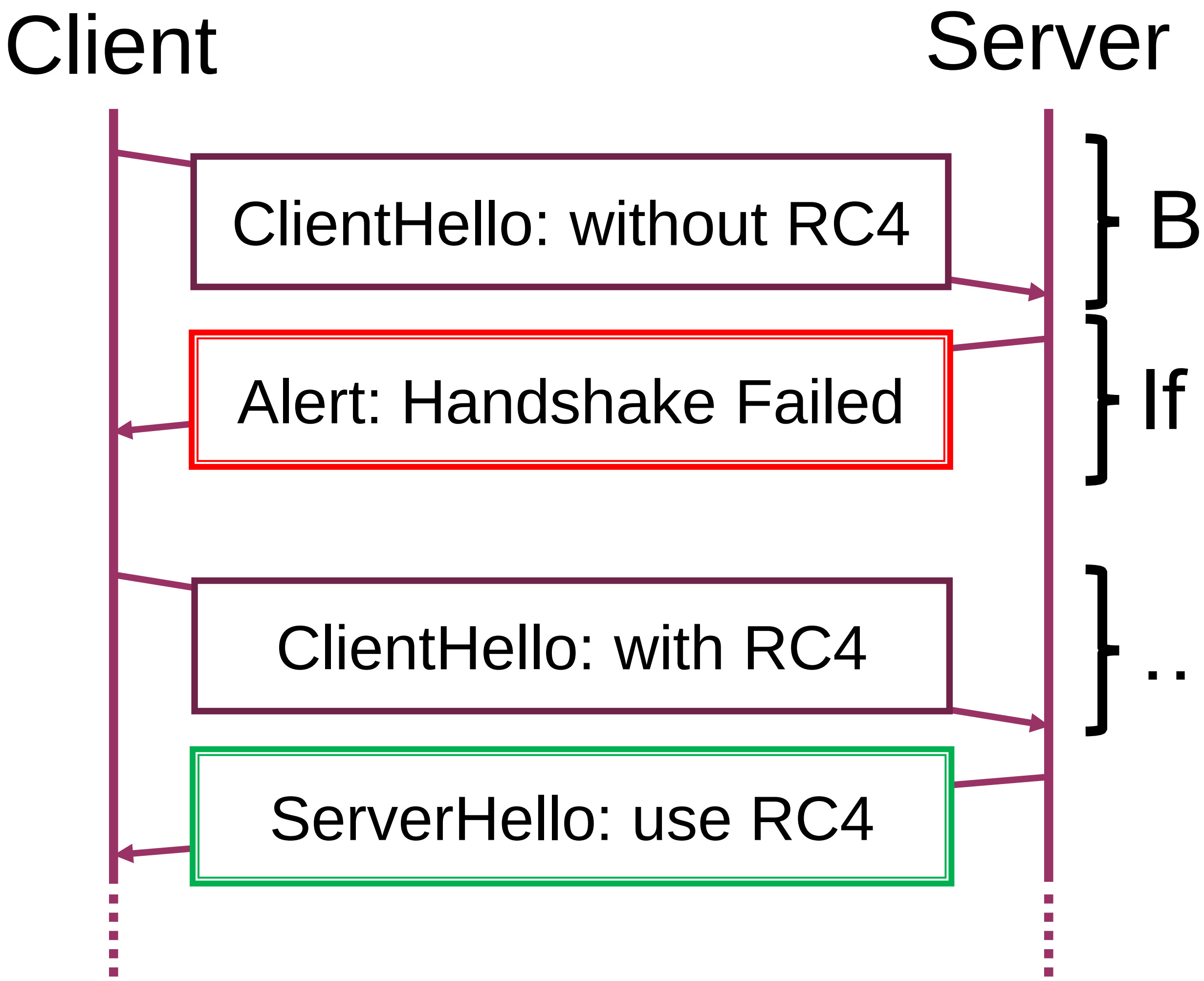
Fallback to RC4



Browser first tries without RC4

If that fails …

Fallback to RC4

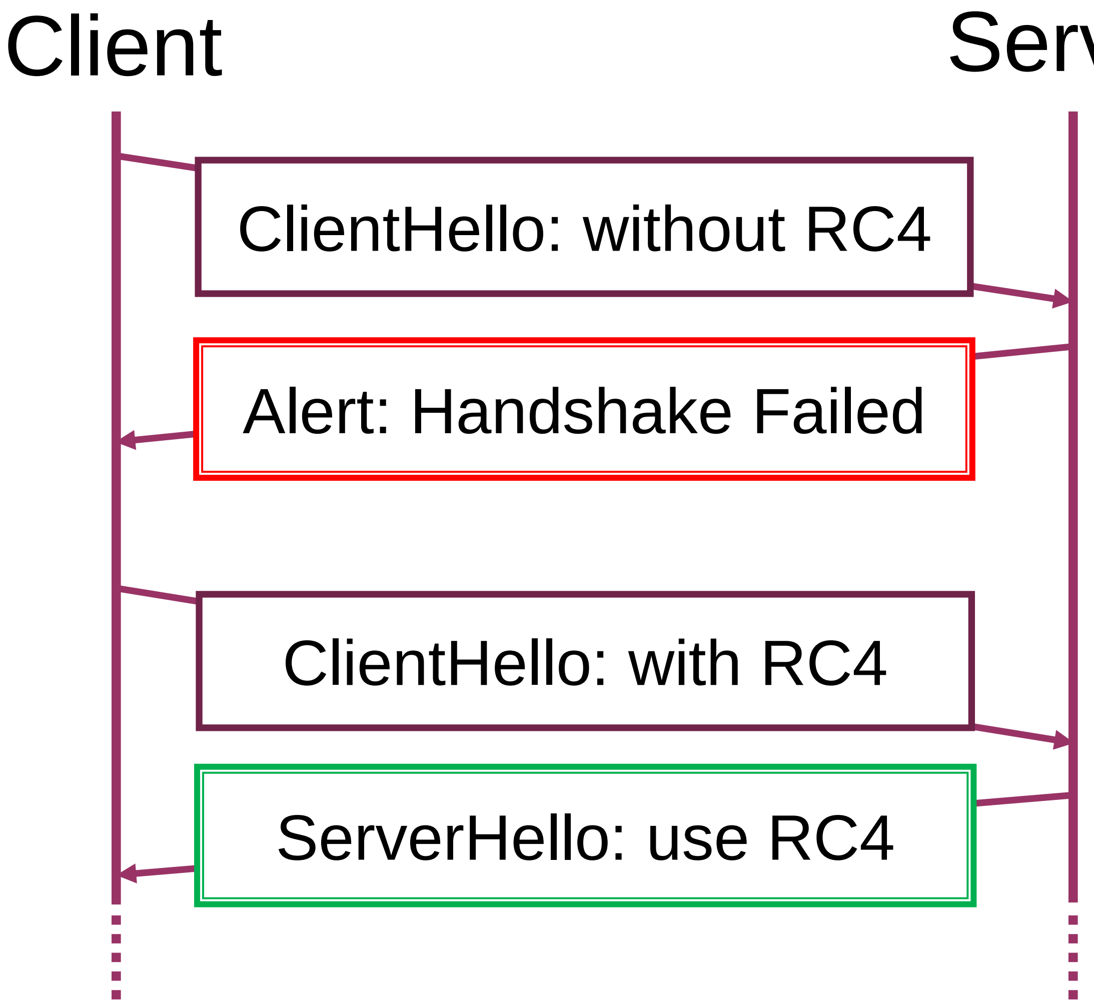


Browser first tries without RC4

If that fails …

… fallback to RC4

Fallback to RC4



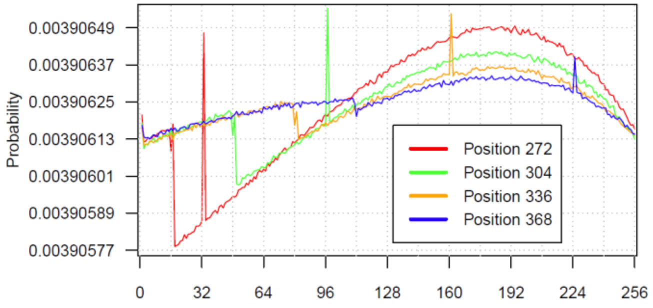
Browser first tries without RC4

Forgeable by attacker!

… fallback to RC4

- **Fallback provides no security**
- But useful to determine how many servers require RC4

Contributions: how did we kill RC4?



New Biases

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

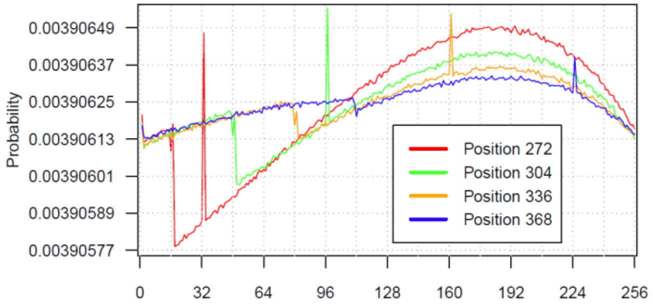
Plaintext Recovery

Break WPA-TKIP

Attack HTTPS

DistriNet
iMinds
KU LEUVEN

Contributions: how did we kill RC4?



New Biases

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

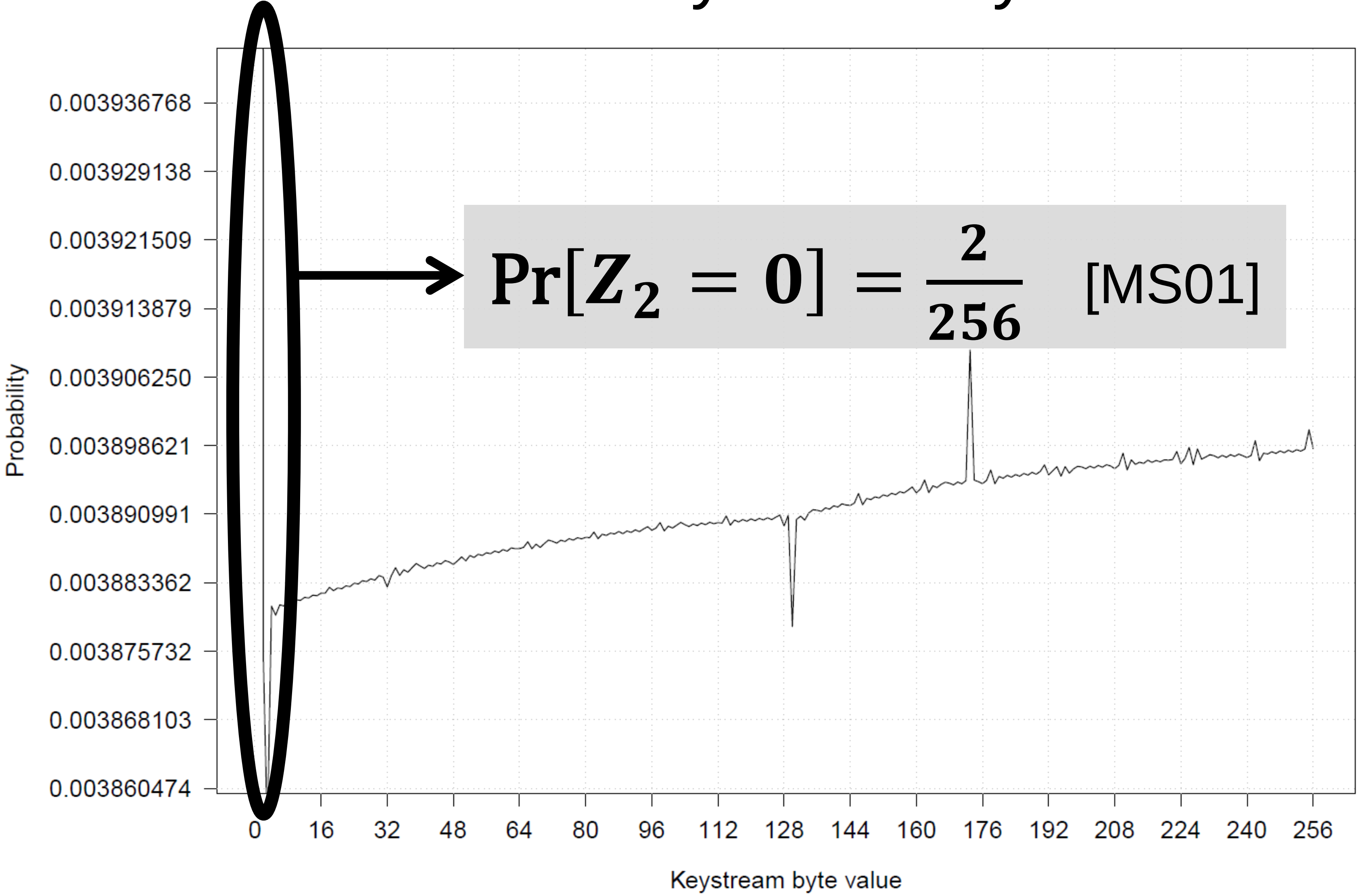
Plaintext Recovery

Break WPA-TKIP

Attack HTTPS

First: Existing Biases

Distribution keystream byte 2



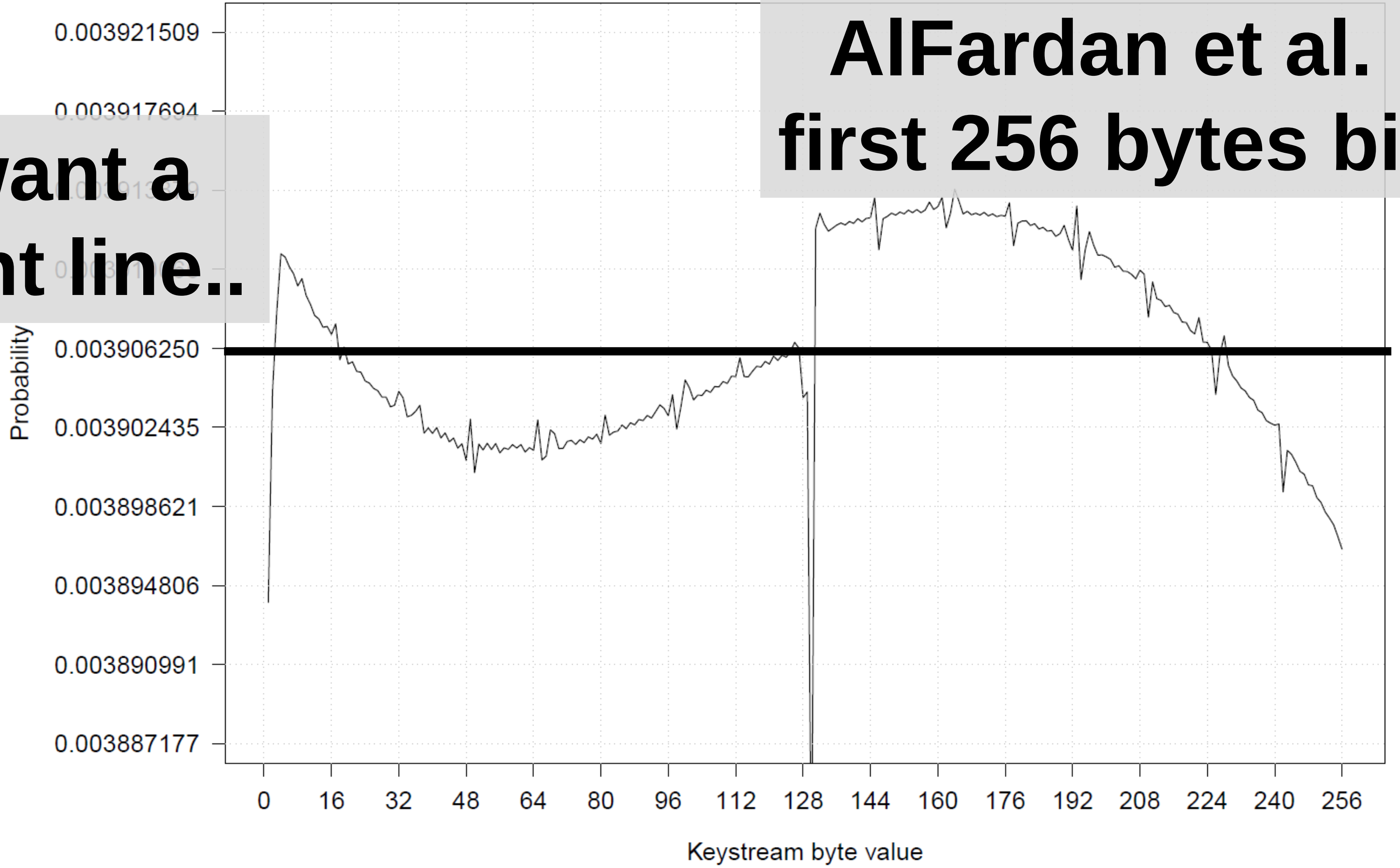
$$\Pr[Z_2 = 0] = \frac{2}{256}$$ [MS01]

DistriNet iMinds KU LEUVEN

First: Existing Biases

Short-term biases

Distribution keystream byte 1 (to 256)



We want a straight line..

AlFardan et al. '13: first 256 bytes biased

DistriNet iMinds KU LEUVEN

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

A	B	S	A	B

Search for new biases

Traditional emperical approach:

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

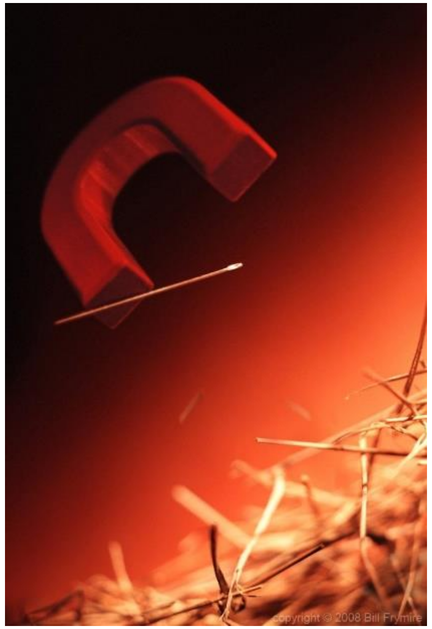
Fluhrer-McGrew biases: only 8 of 65 536 pairs are biased

How to automate the search?

Search for new biases

Traditional emperical approach:

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



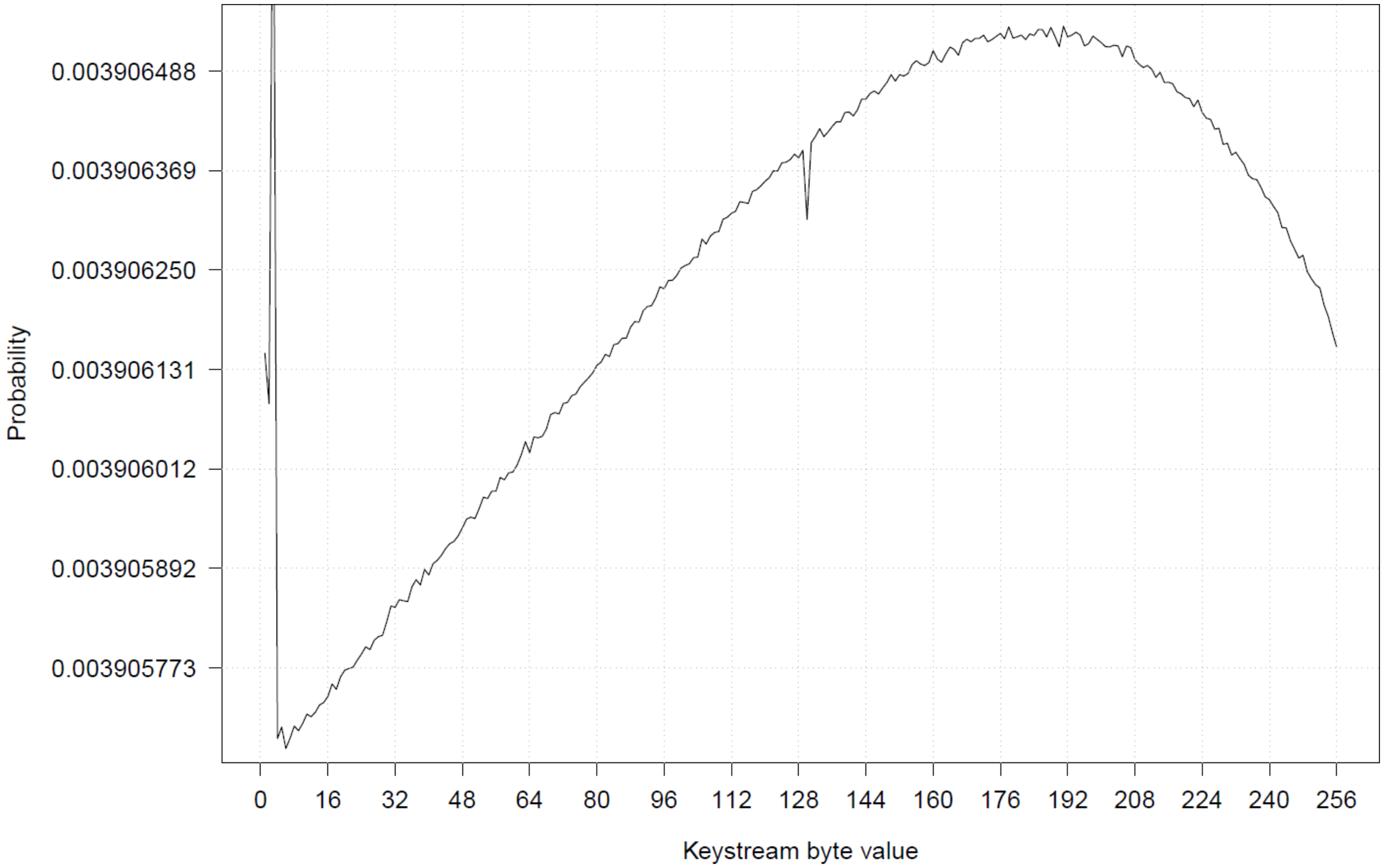
Hypothesis tests!

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

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

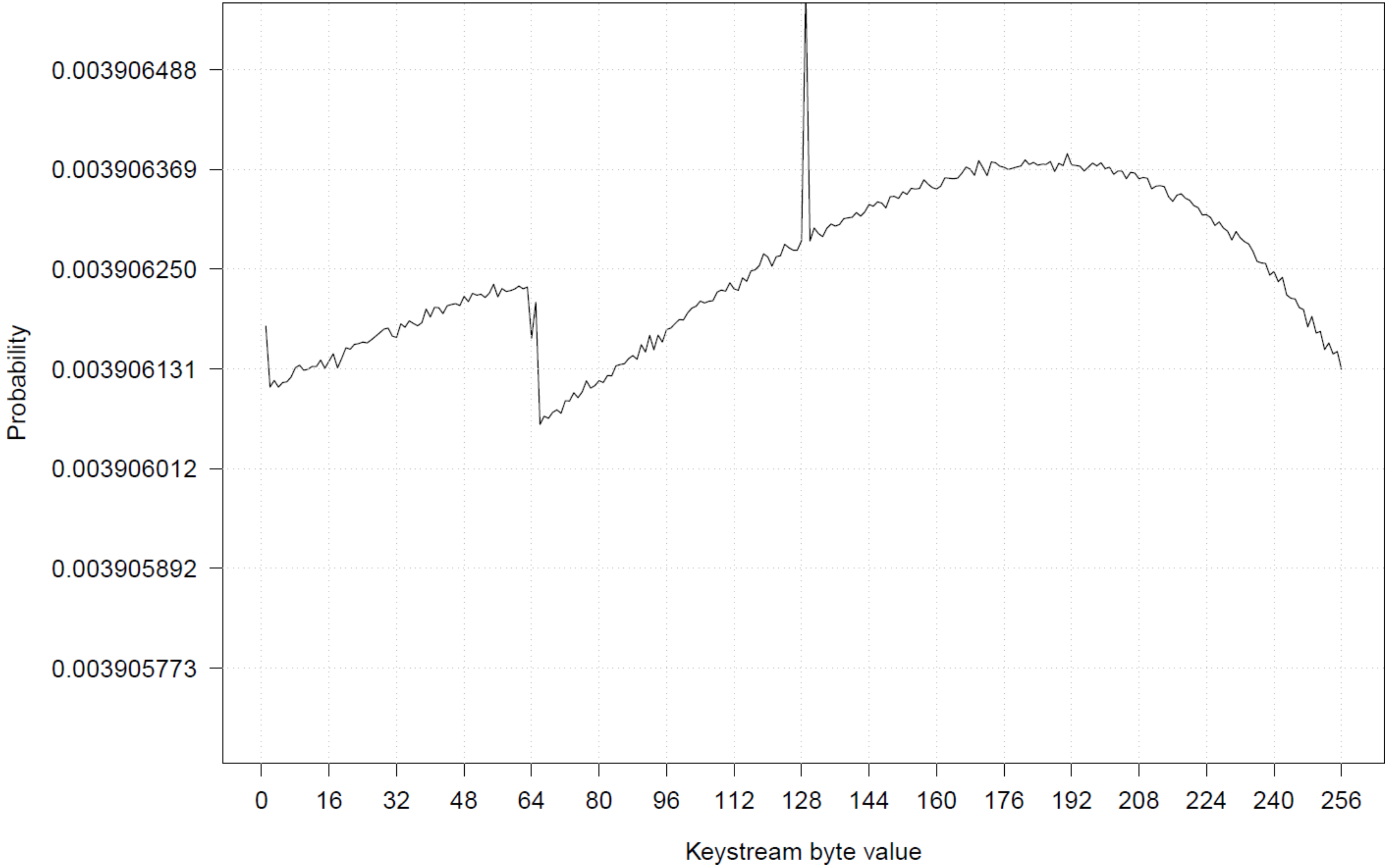
Biases in Bytes 258-513

Example: keystream byte 258



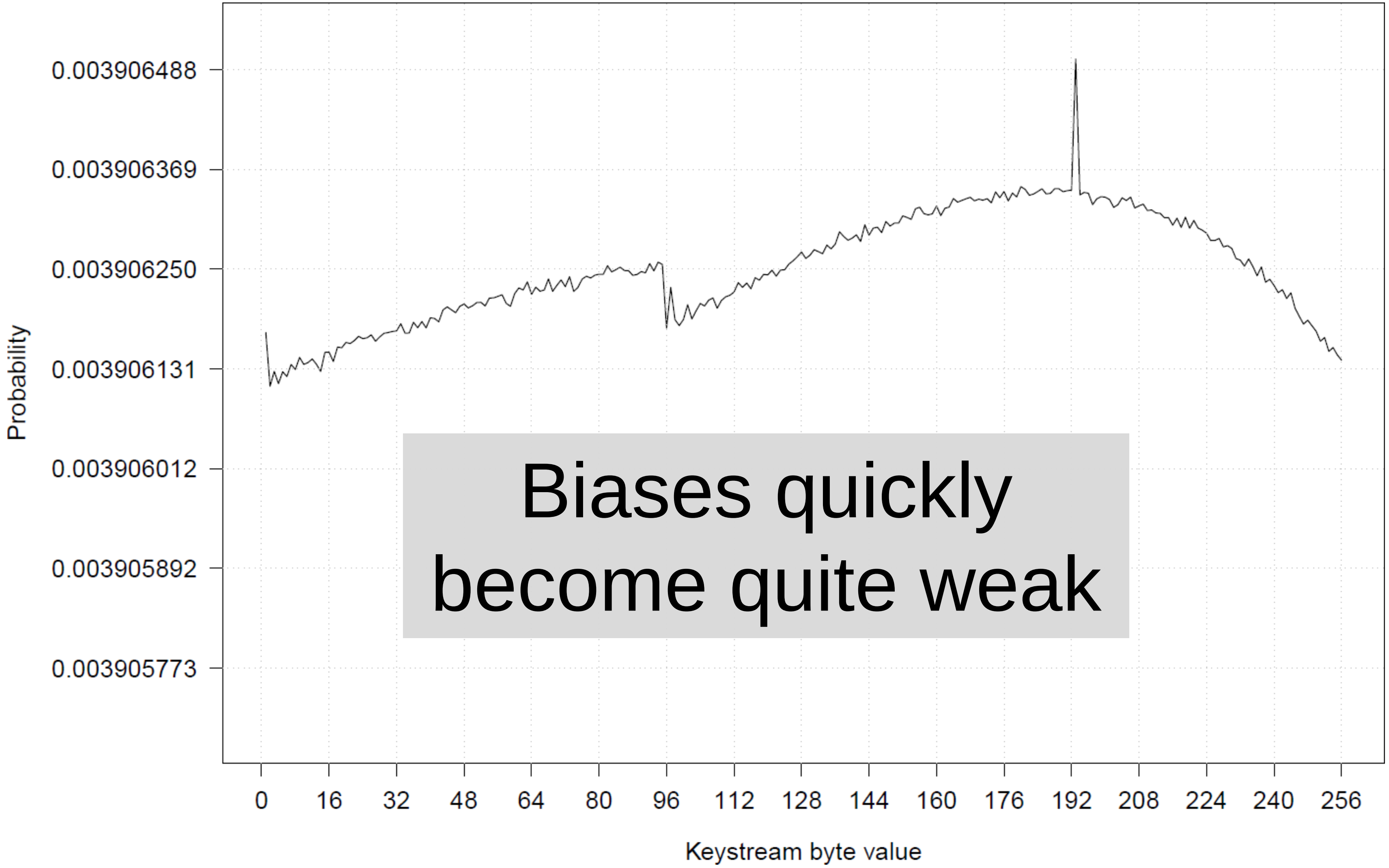
Biases in Bytes 258-513

Example: keystream byte 320



Biases in Bytes 258-513

Example: keystream byte 352



New Long-term Bias

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

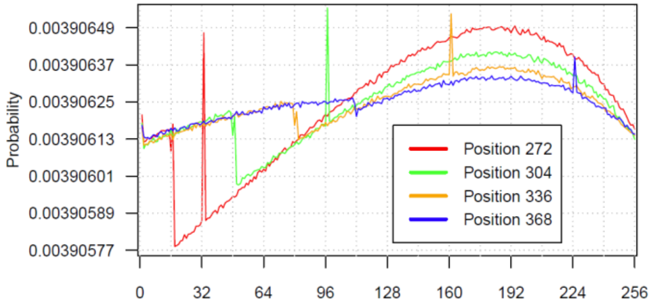
with probability $2^{-16}(1 + 2^{-8})$

Every block of 256 bytes

Additional Biases

See paper!

Contributions: how did we kill RC4?



New Biases

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

Plaintext Recovery

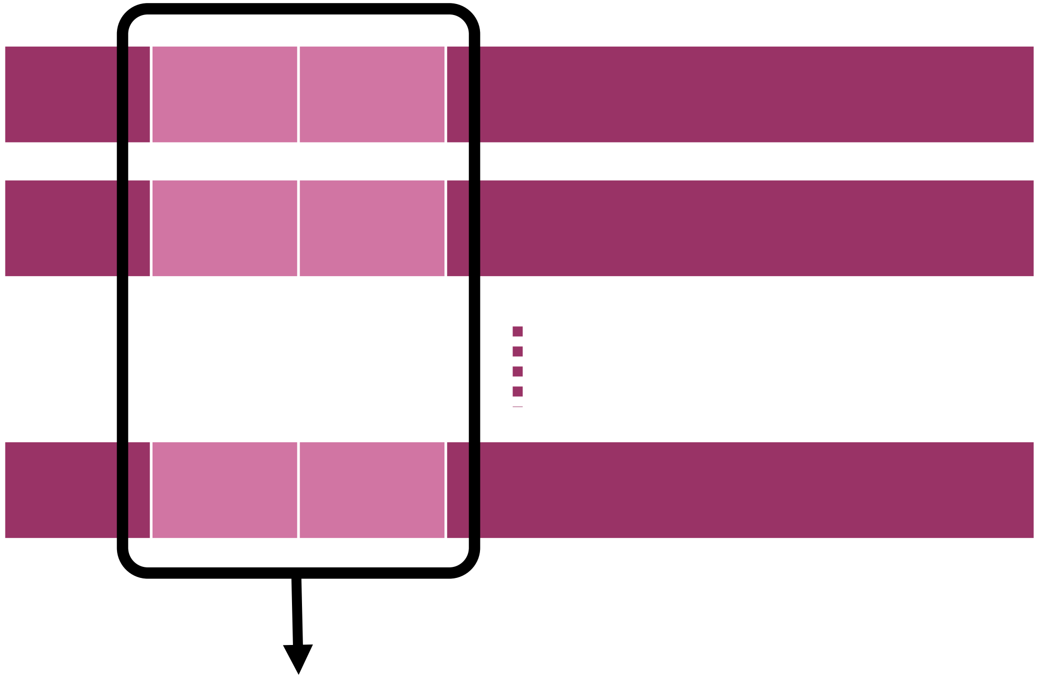
Break WPA-TKIP

Attack HTTPS

DistriNet iMinds KU LEUVEN

Existing Methods [AlFardan et al. ‘13]

Plaintext encrypted under several keystreams

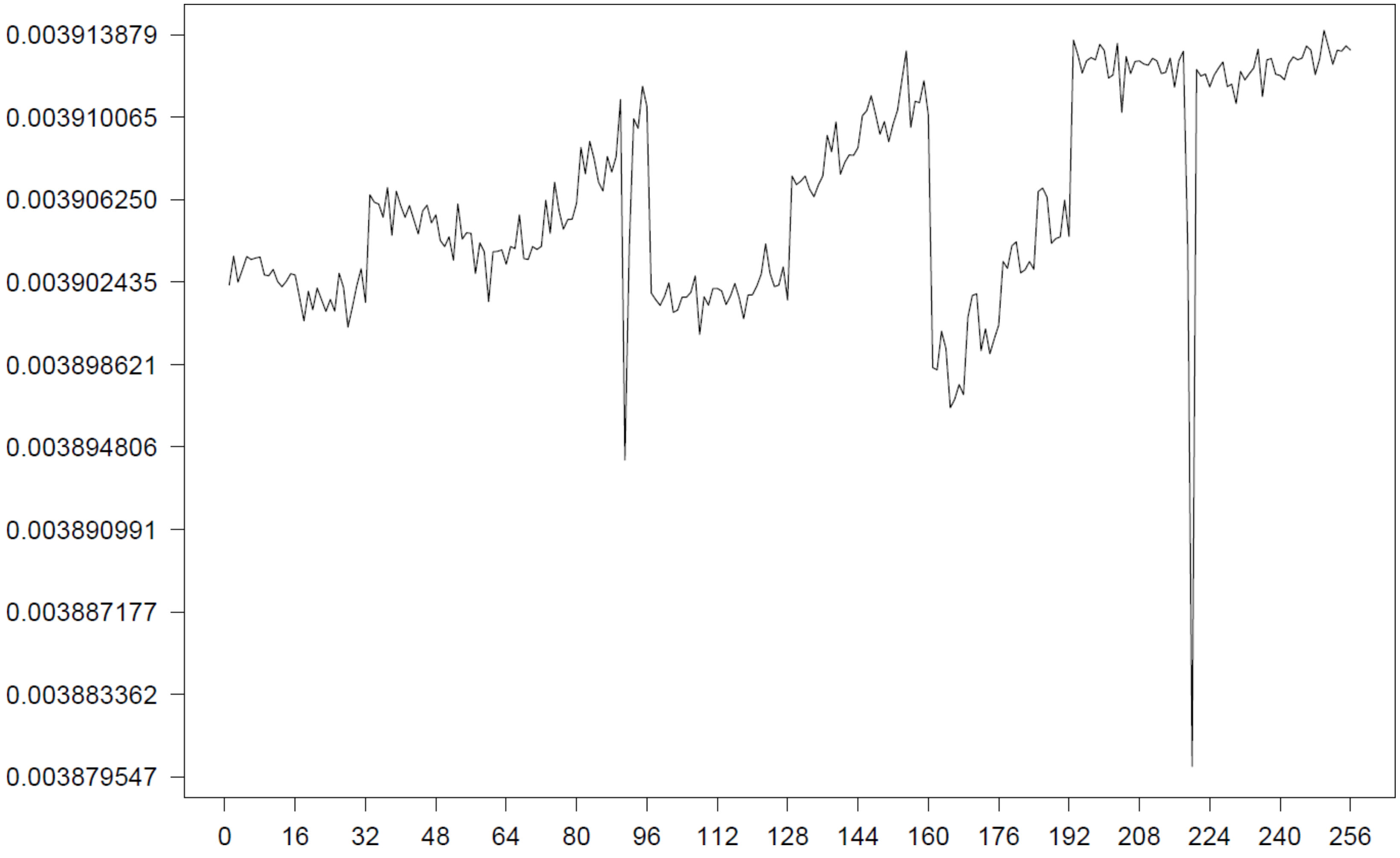


Verify guess: how close to real keystream distribution?

Ciphertext Distribution $\oplus$ Plaintext guess μ $=$ **Induced** keystream distribution

Example: Decrypt byte 1

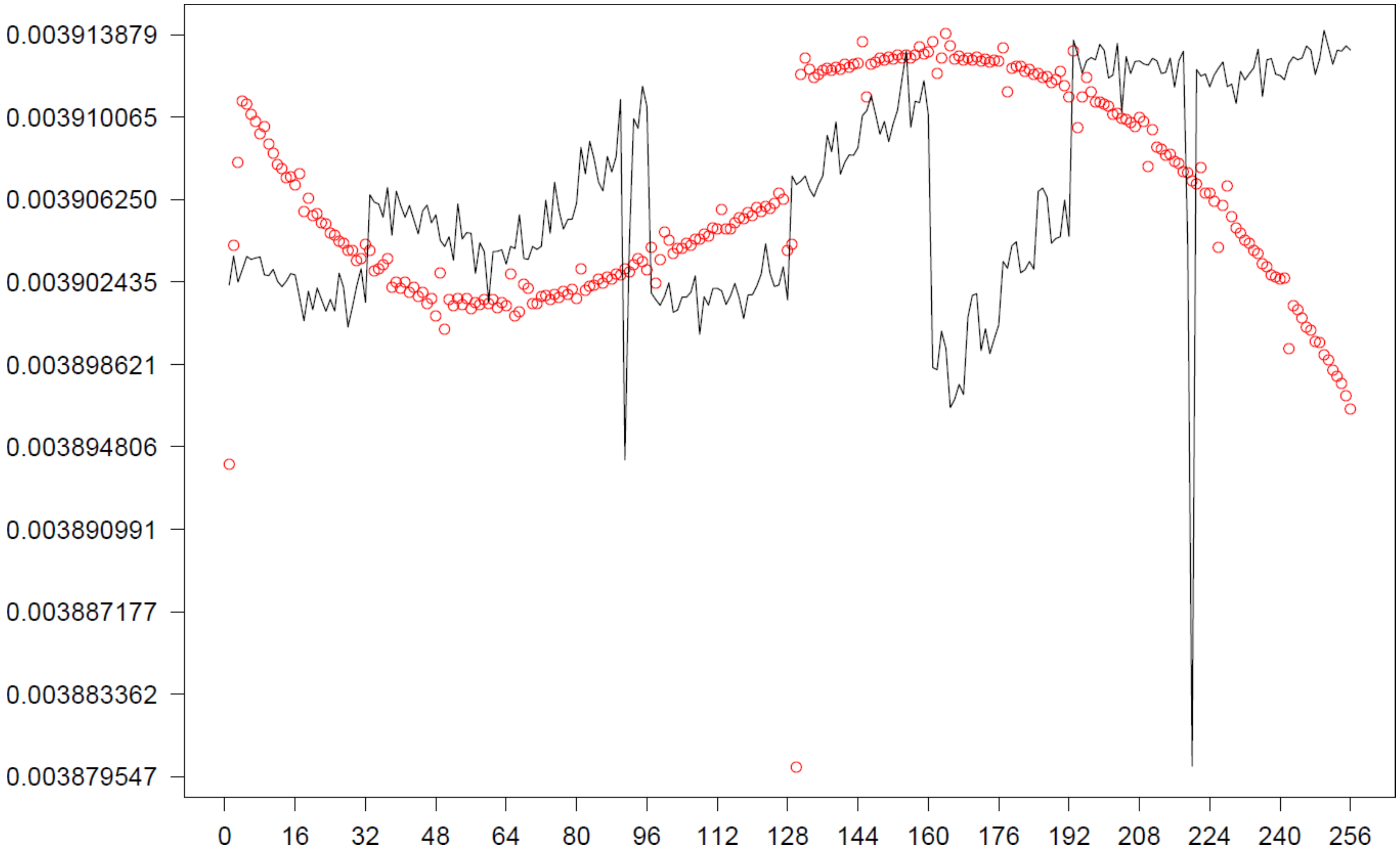
Ciphertext Distribution



DistriNet
iMinds
KU LEUVEN

Example: Decrypt byte 1

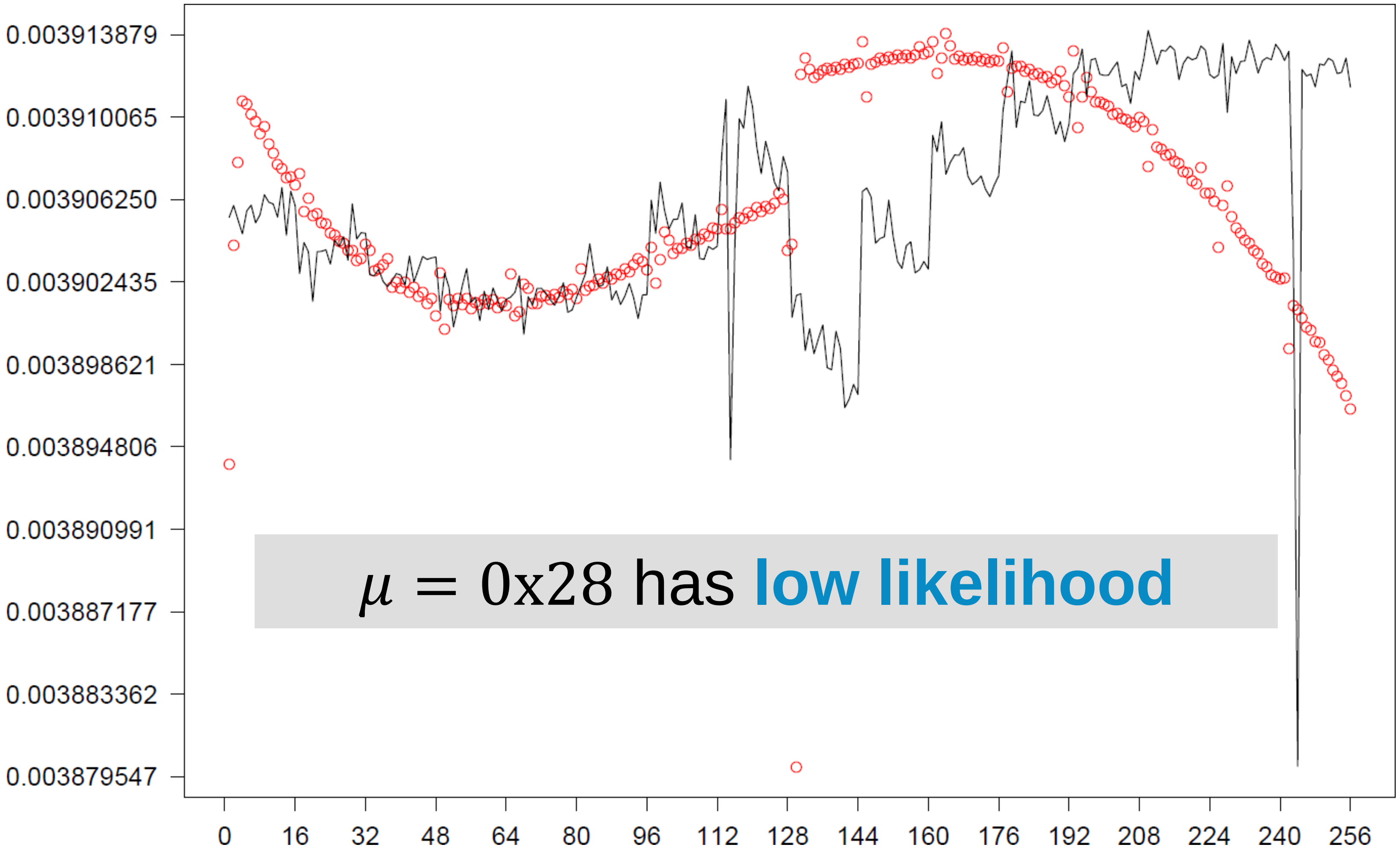
RC4 & Ciphertext distribution



DistriNet
iMinds KU LEUVEN

Example: Decrypt byte 1

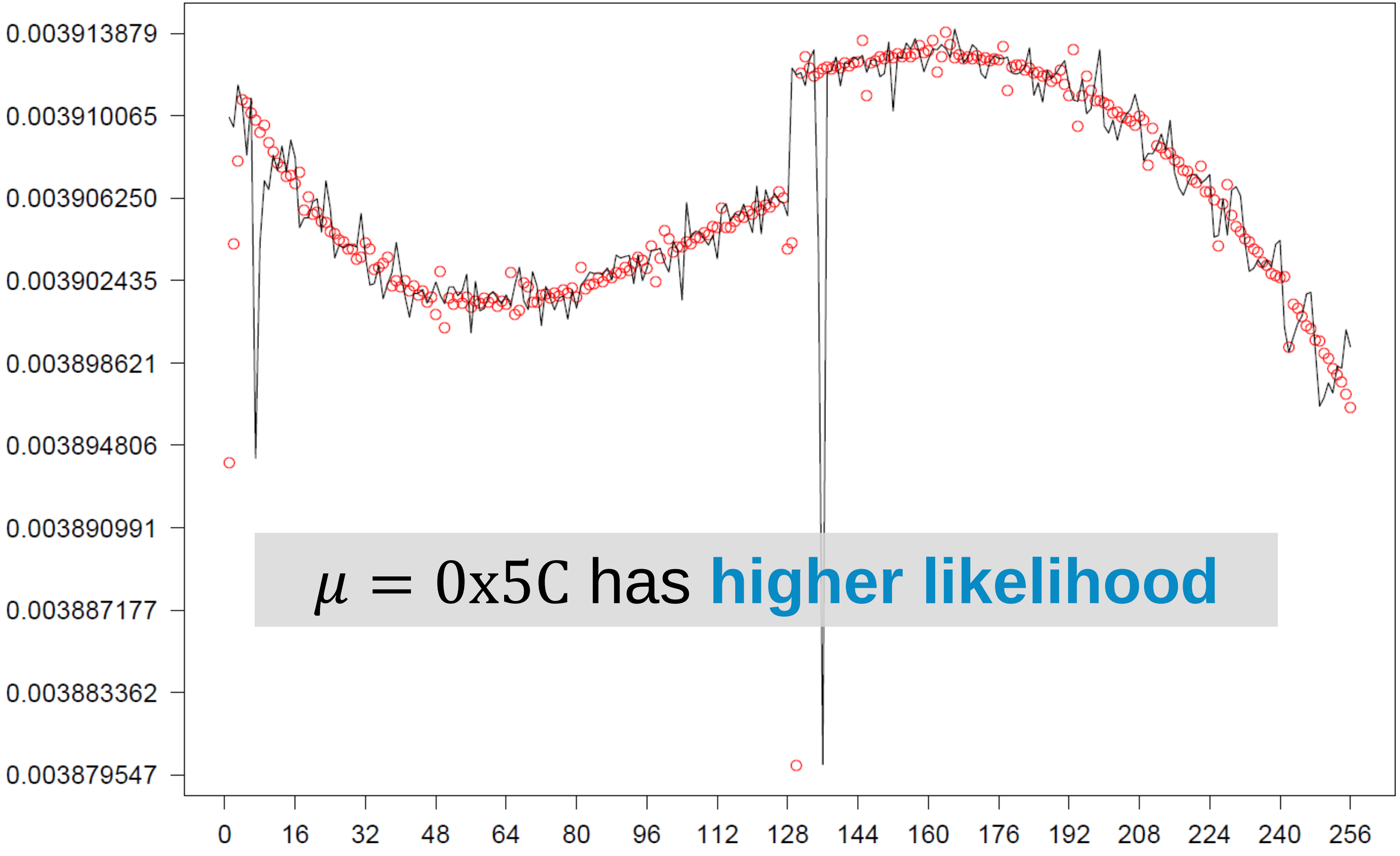
If plaintext byte $\mu = 0x28$: **RC4** & **Induced**



DistriNet iMinds KU LEUVEN

Example: Decrypt byte 1

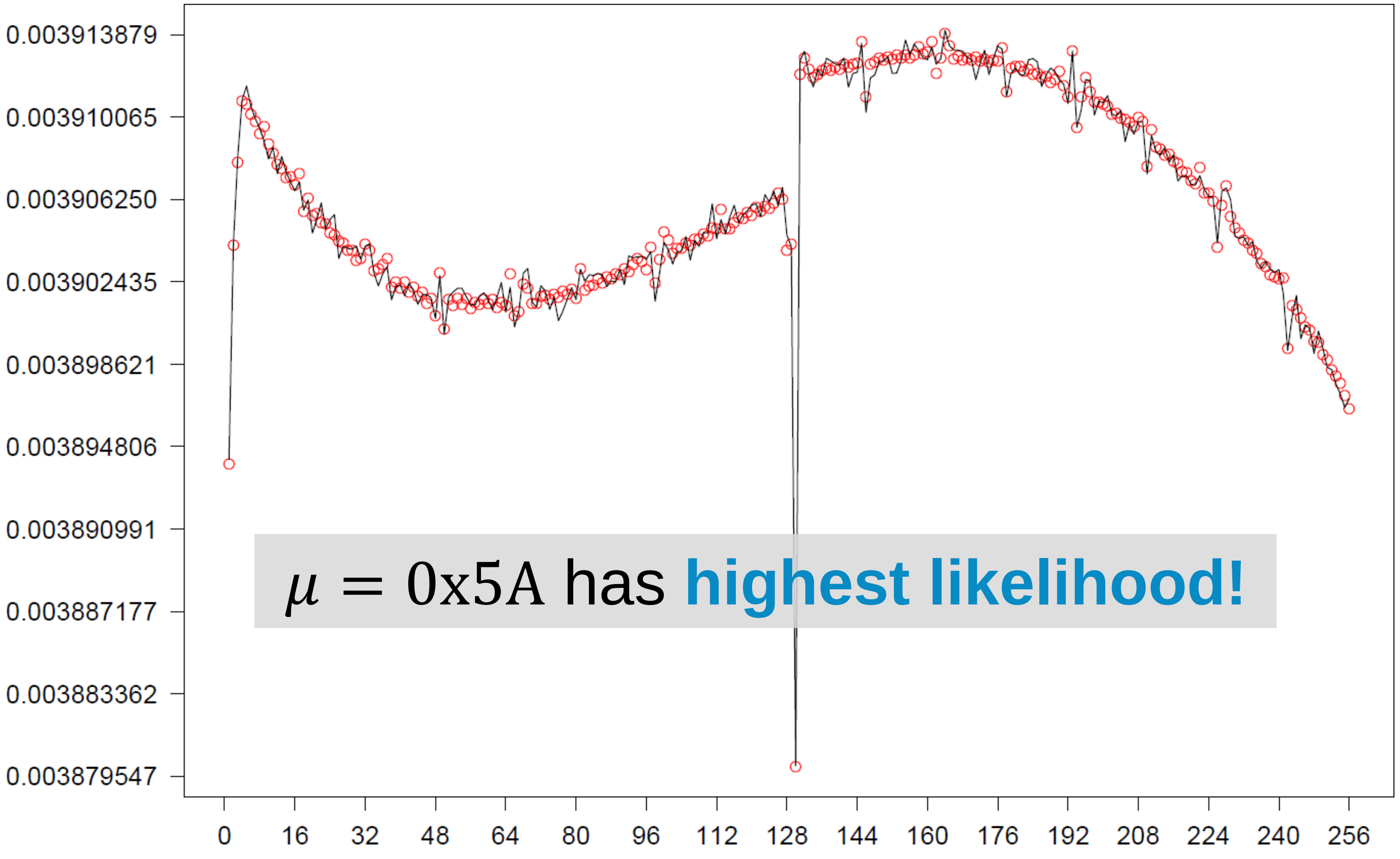
If plaintext byte $\mu = \mathrm{0x5C}$: **RC4** & **Induced**



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

Example: Decrypt byte 1

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



Types of likelihood estimates

Previous works: pick value with highest likelihood.

Better idea: list of candidates in decreasing likelihood:

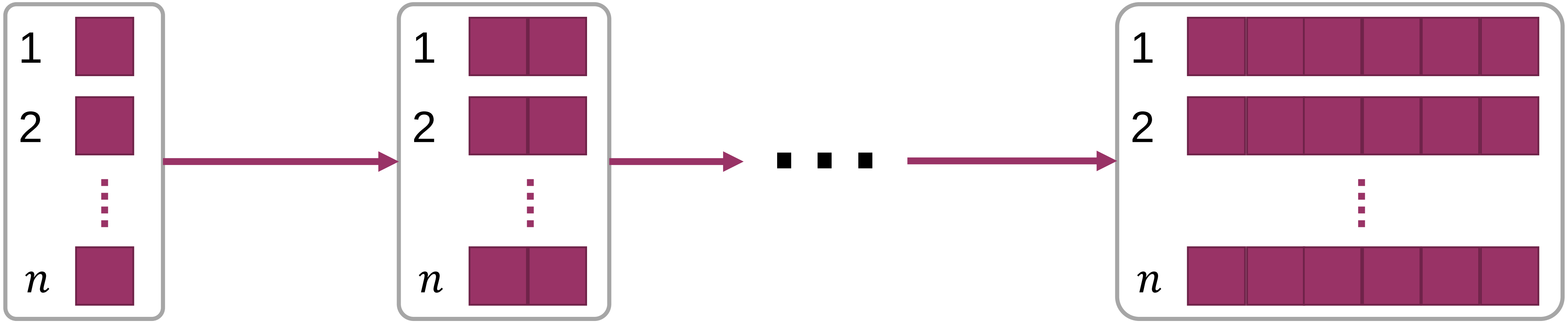
- Most likely one may not be correct!
- Prune bad candidates (e.g. bad CRC)
- Brute force cookies or passwords

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

Known Plaintext

Unknown Plaintext

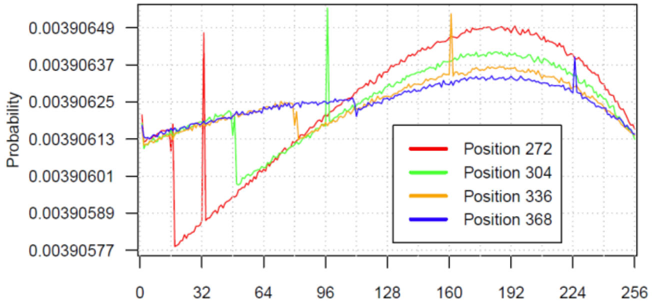
Assume there’s **surrounding known plaintext**!

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

Contributions: how did we 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

TKIP Background

How are packets sent/received?

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

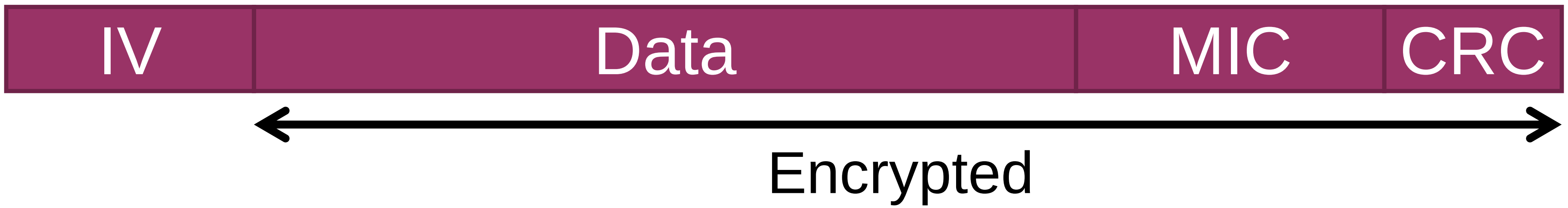
How are packets sent/received?

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

DistriNet
iMinds
KU LEUVEN

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)

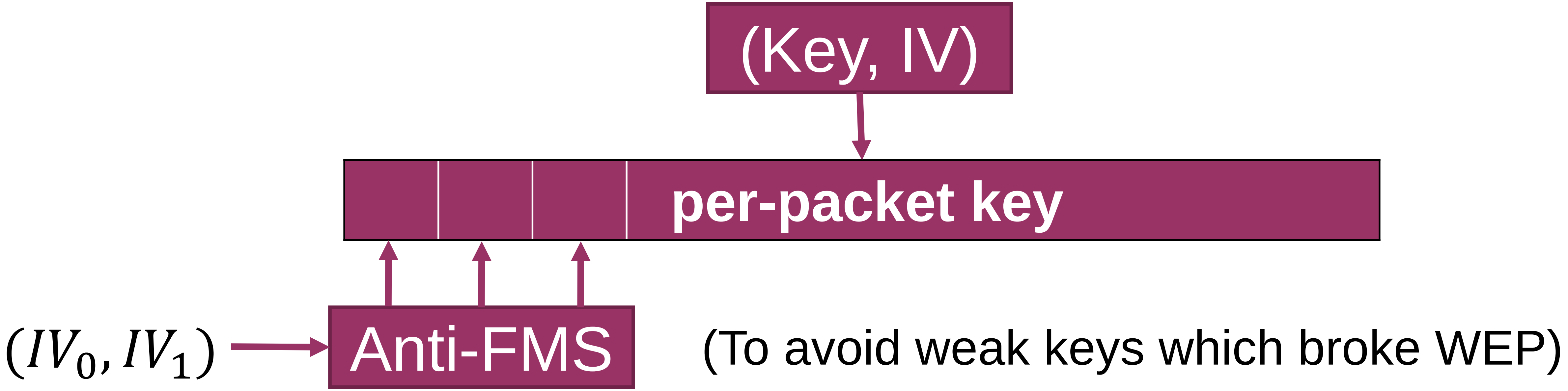
Flaw #1: TKIP Per-packet Key

(Key, IV)

↓

per-packet key

Flaw #1: TKIP Per-packet Key



→ IV-dependent biases in keystream

[Gupta/Paterson et al.]

DistriNet iMinds KU LEUVEN

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:

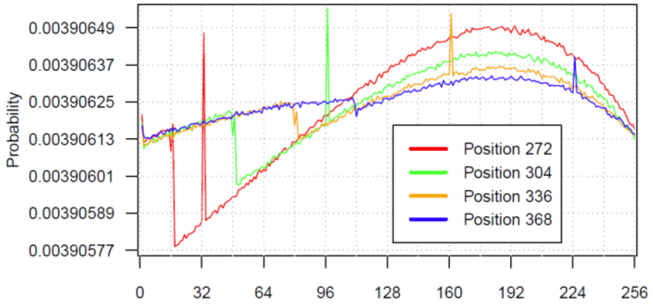
- Need $\approx 2^{24}$ captures to decrypt with high success rates

Emperical tests:

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

DistriNet iMinds KU LEUVEN

Contributions: how did we kill RC4?



New Biases

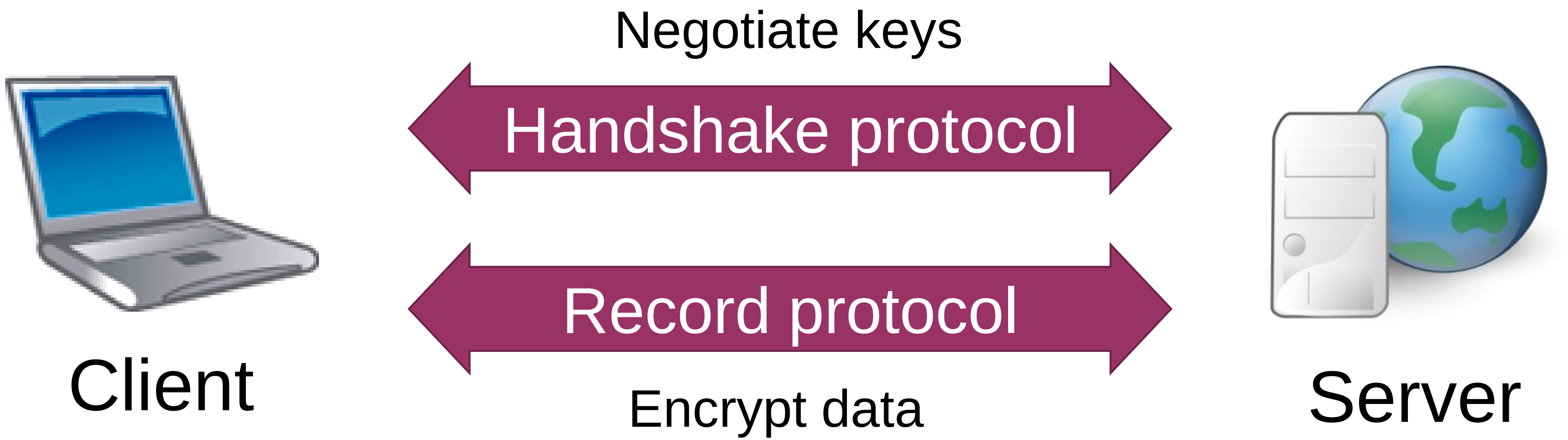
$$\lambda_{\widehat{\mu}} = (1 - \alpha(g))^{|C| - |\widehat{u}|} \cdot \alpha(g)^{|\widehat{\mu}|}$$

Plaintext Recovery

Break WPA-TKIP

Attack HTTPS

TLS Background



→ Focus on **record protocol with RC4** as cipher

DistriNet iMinds KU LEUVEN

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

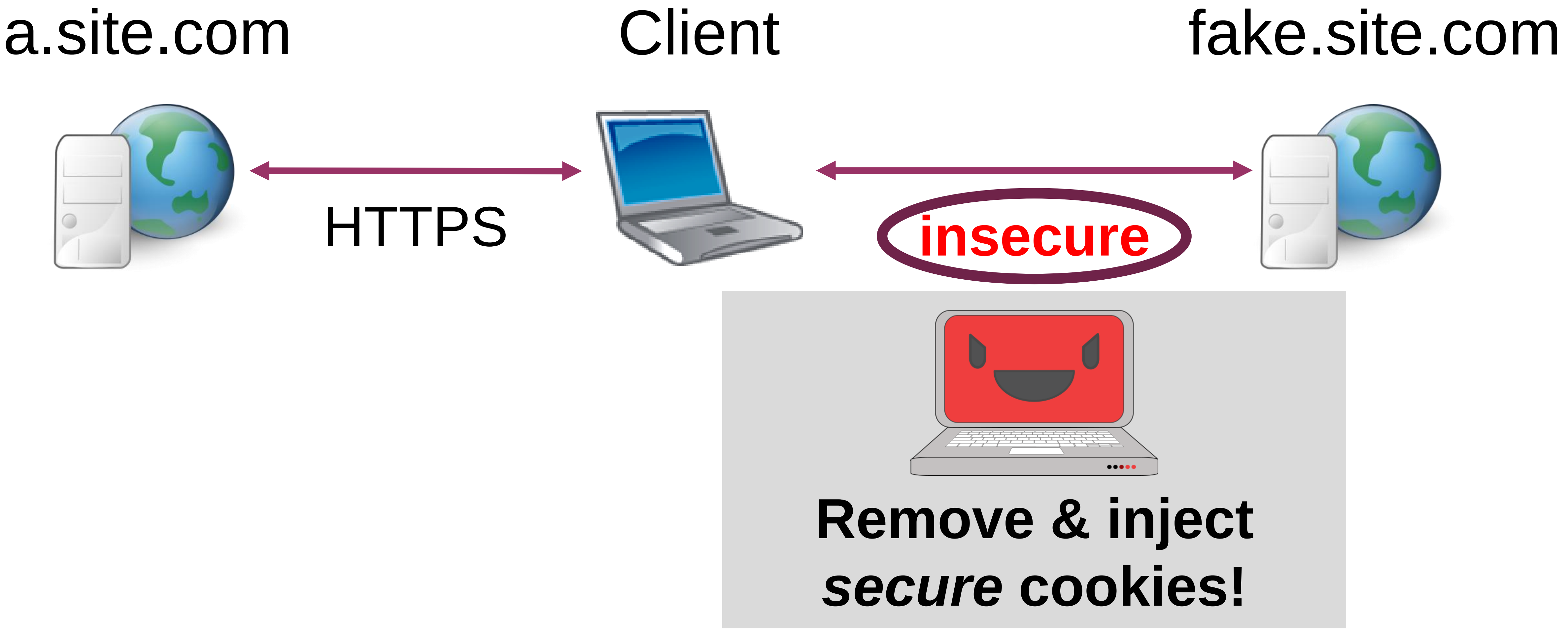
Example: manipulated HTTP request

```
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=aaaaaaaaaaaaaaaa
```

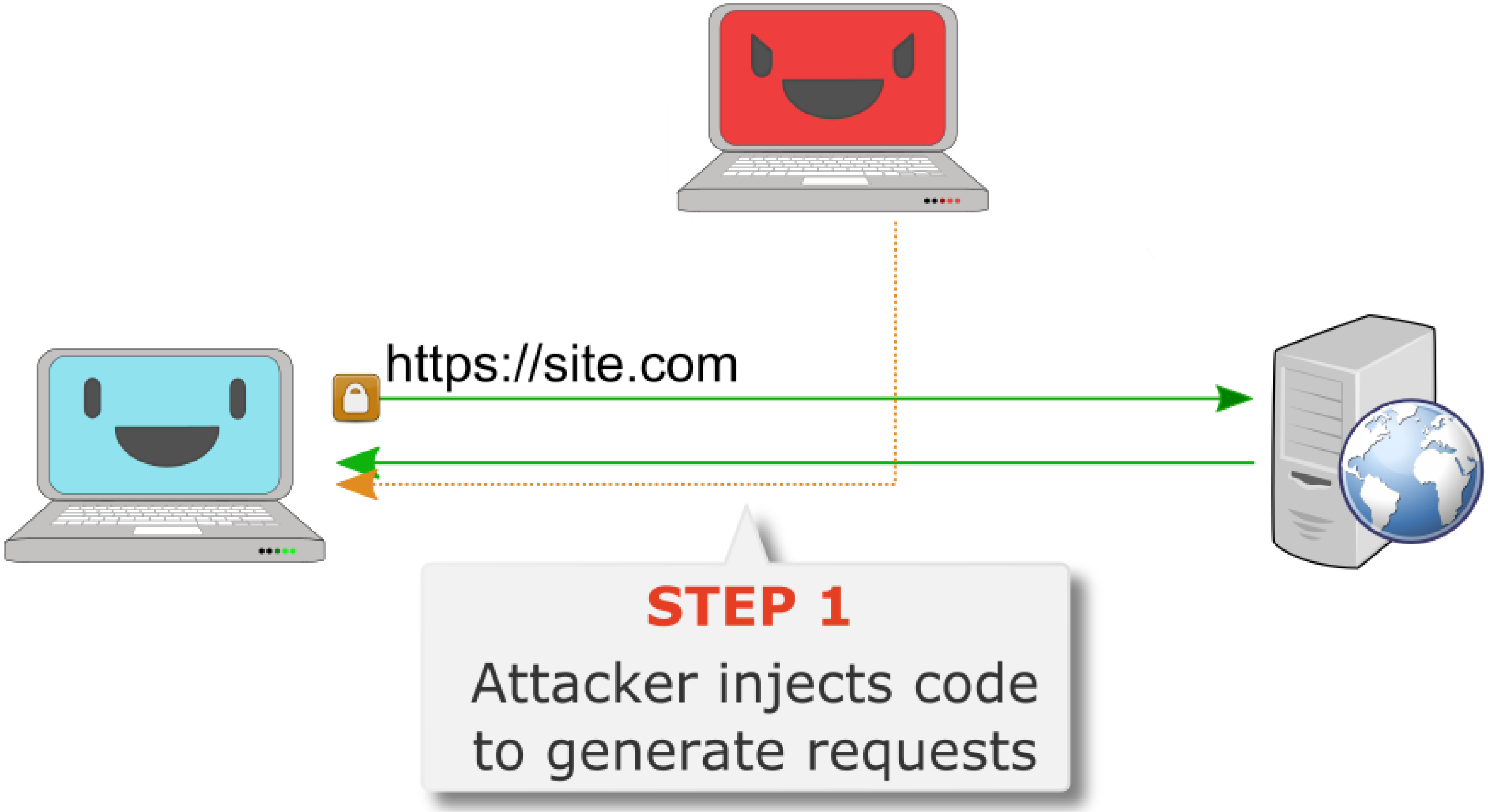
Headers are predictable

Surrounded by known plaintext at both sides

Preparation: manipulating cookies

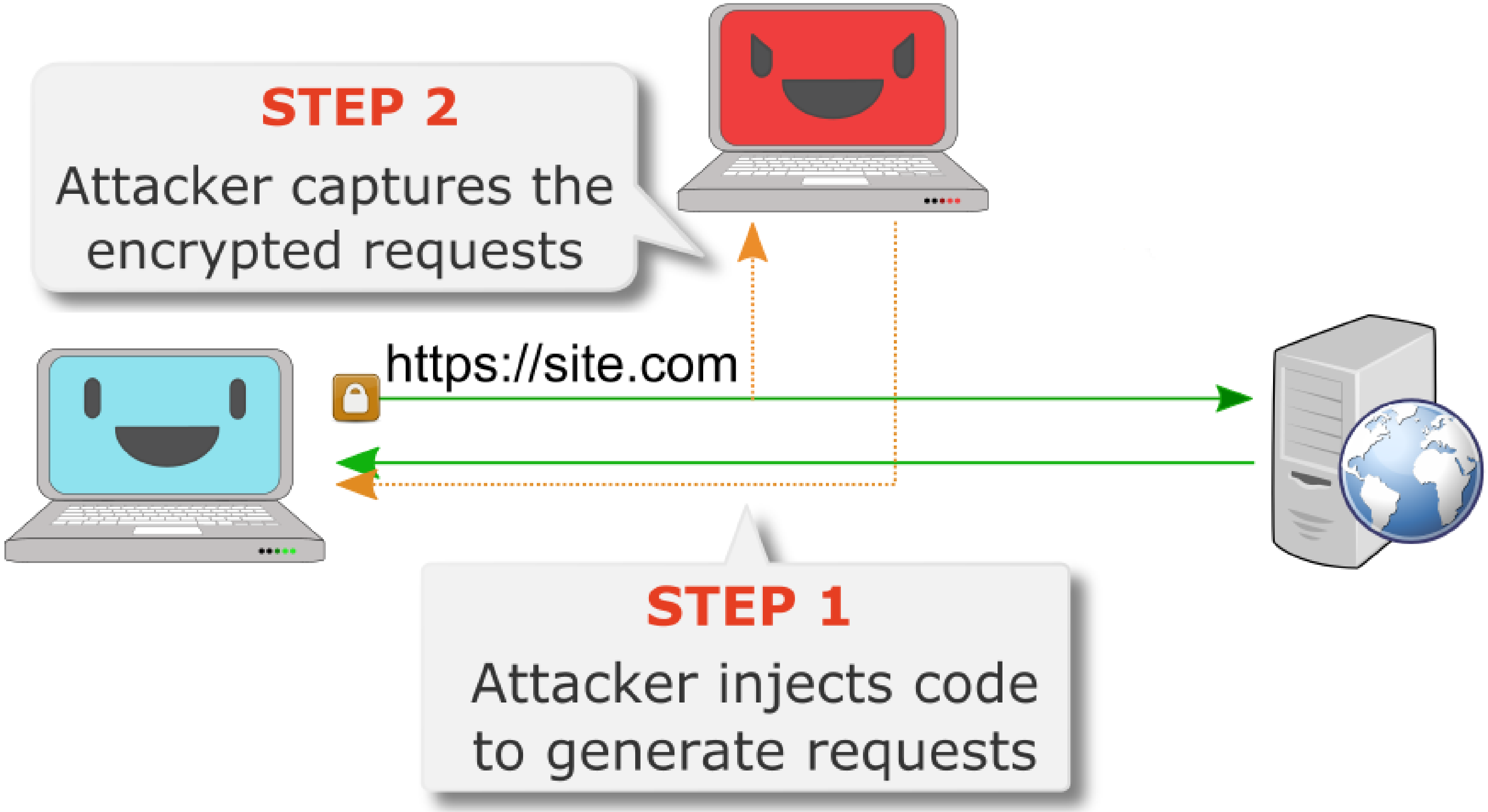


Performing the attack!



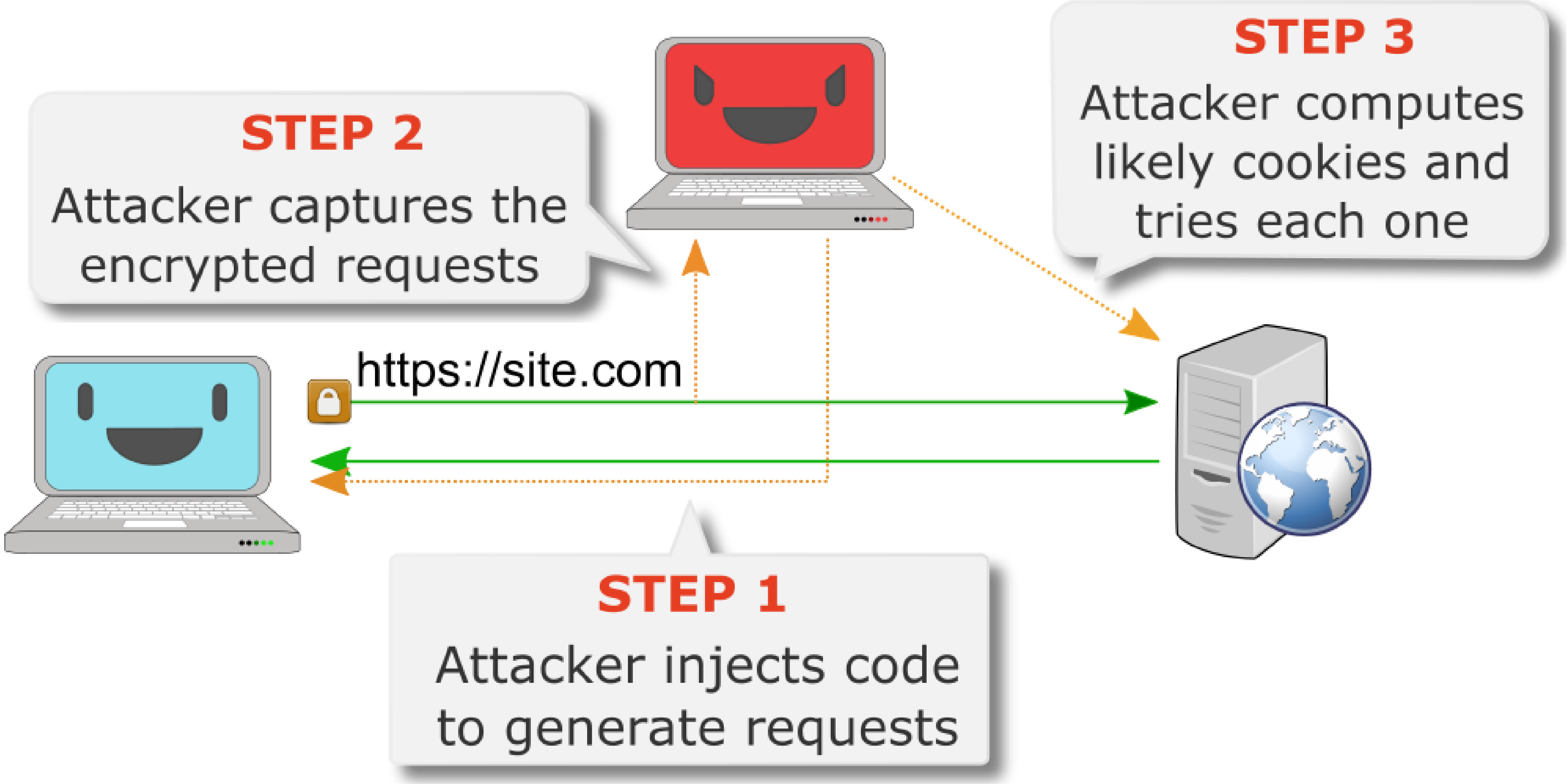
JavaScript: Cross-Origin requests in WebWorkers

Performing the attack!



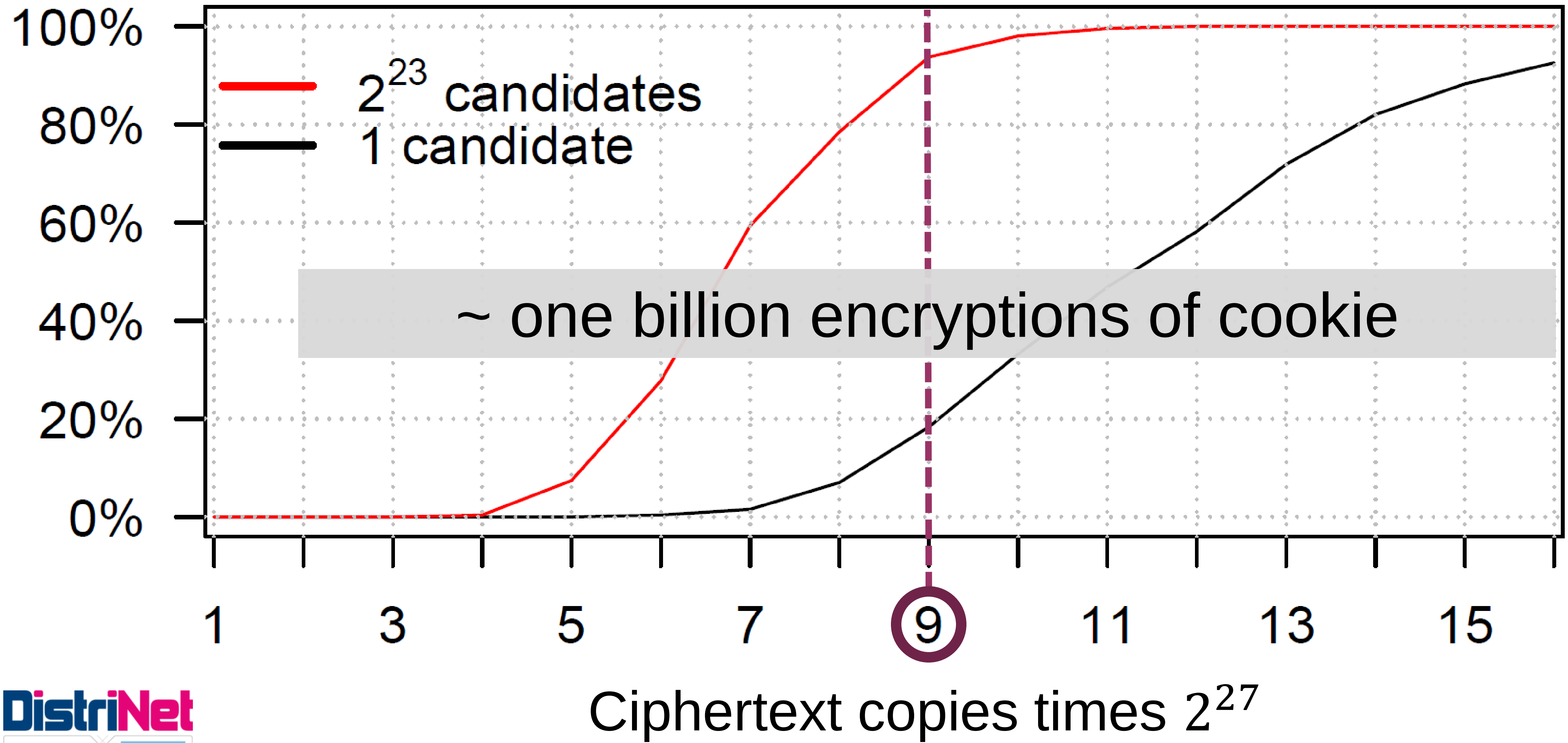
Keep-Alive connection to generate them fast

Performing the attack!

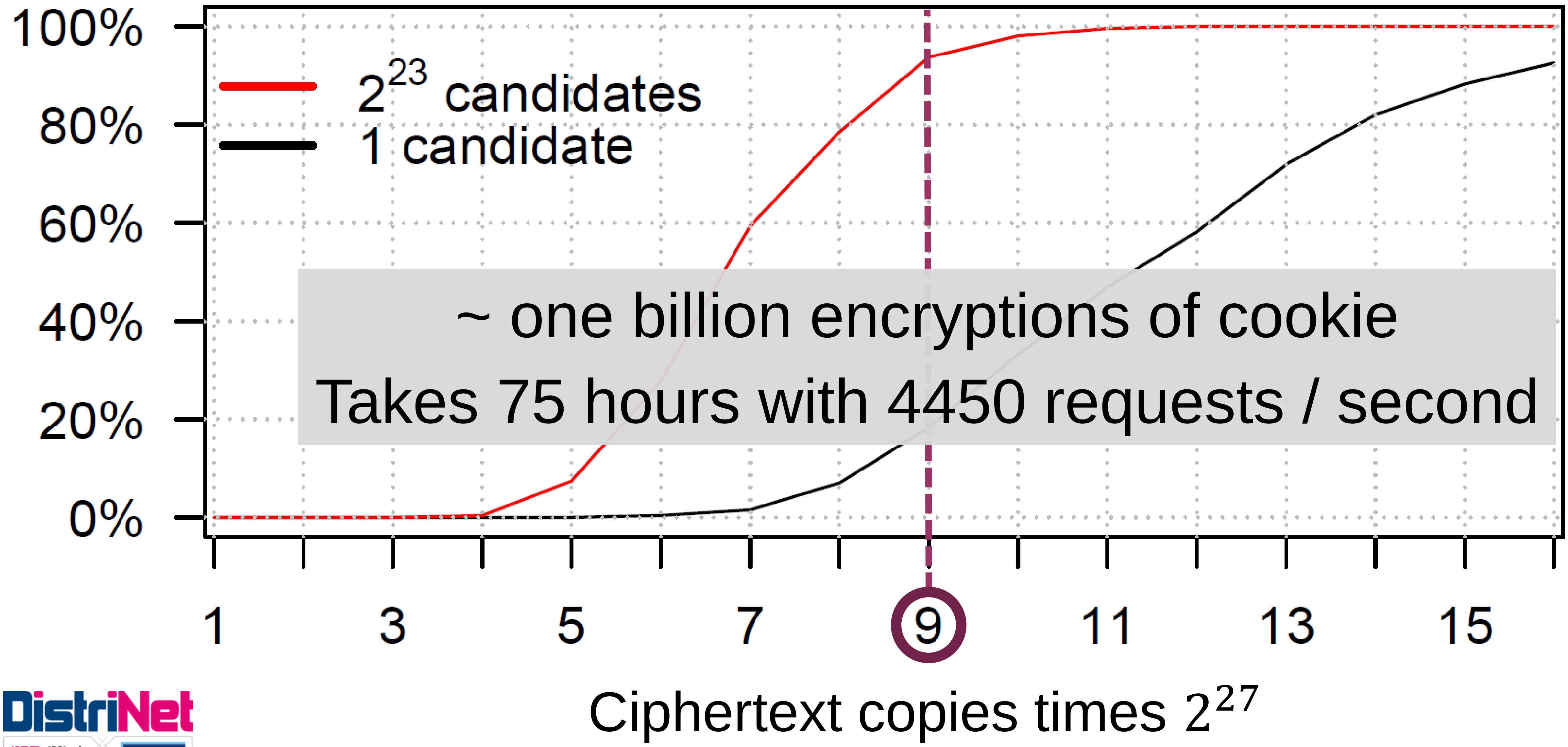


Combine Fluhrer-McGrew and ABSAB biases

Decrypting 16-character cookie



Decrypting 16-character cookie



DistriNet
iMinds KU LEUVEN

Decrypting 16-character cookie

DEMO!

rc4nomore.com

Questions?

May the bias be ever in your favor

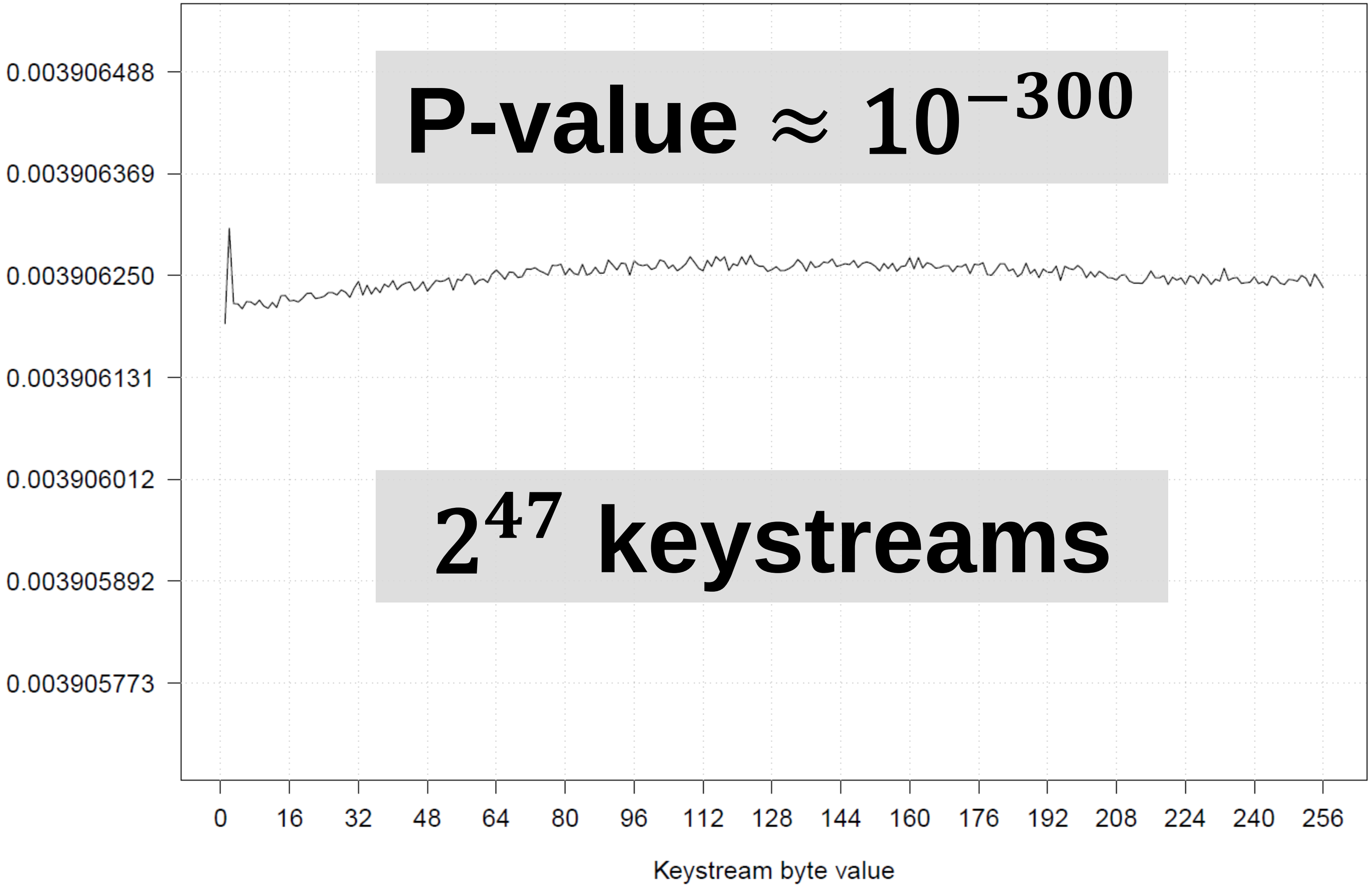
DistriNet
iMinds KU LEUVEN

Questions?

May the bias be ever in your favor

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

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

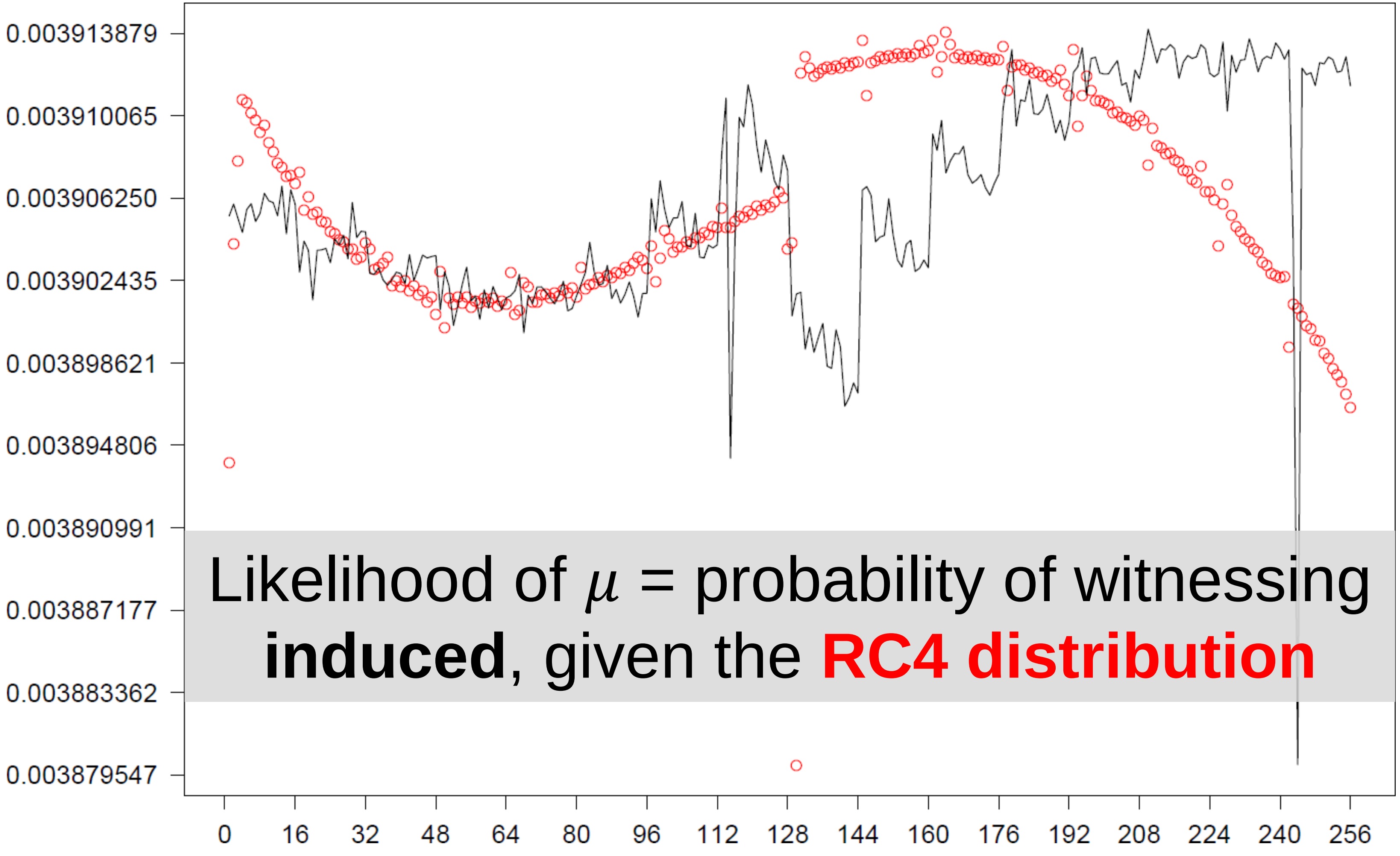
Keystream bytes Z_1 and Z_2

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

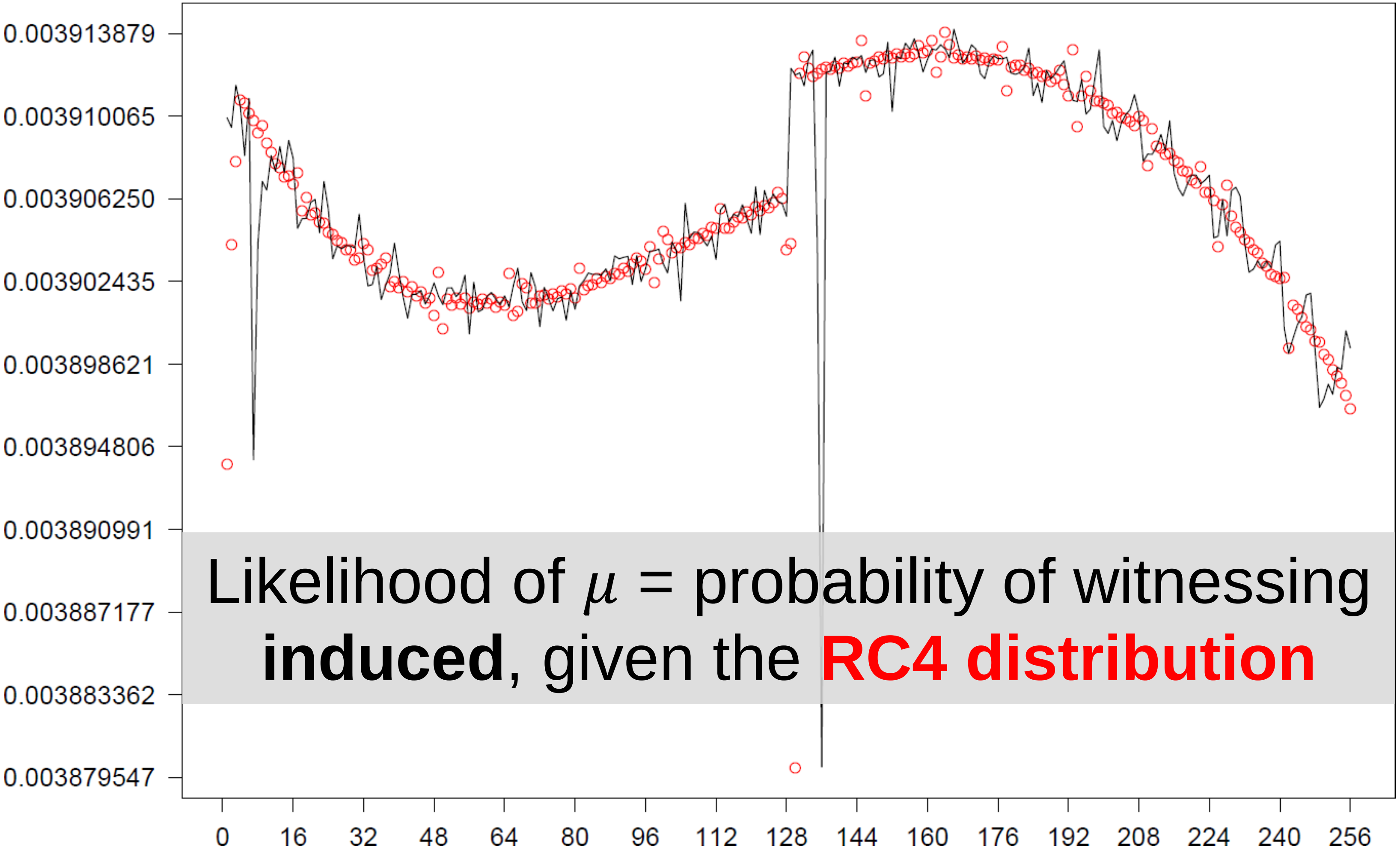
Example: Decrypt byte 1

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



Example: Decrypt byte 1

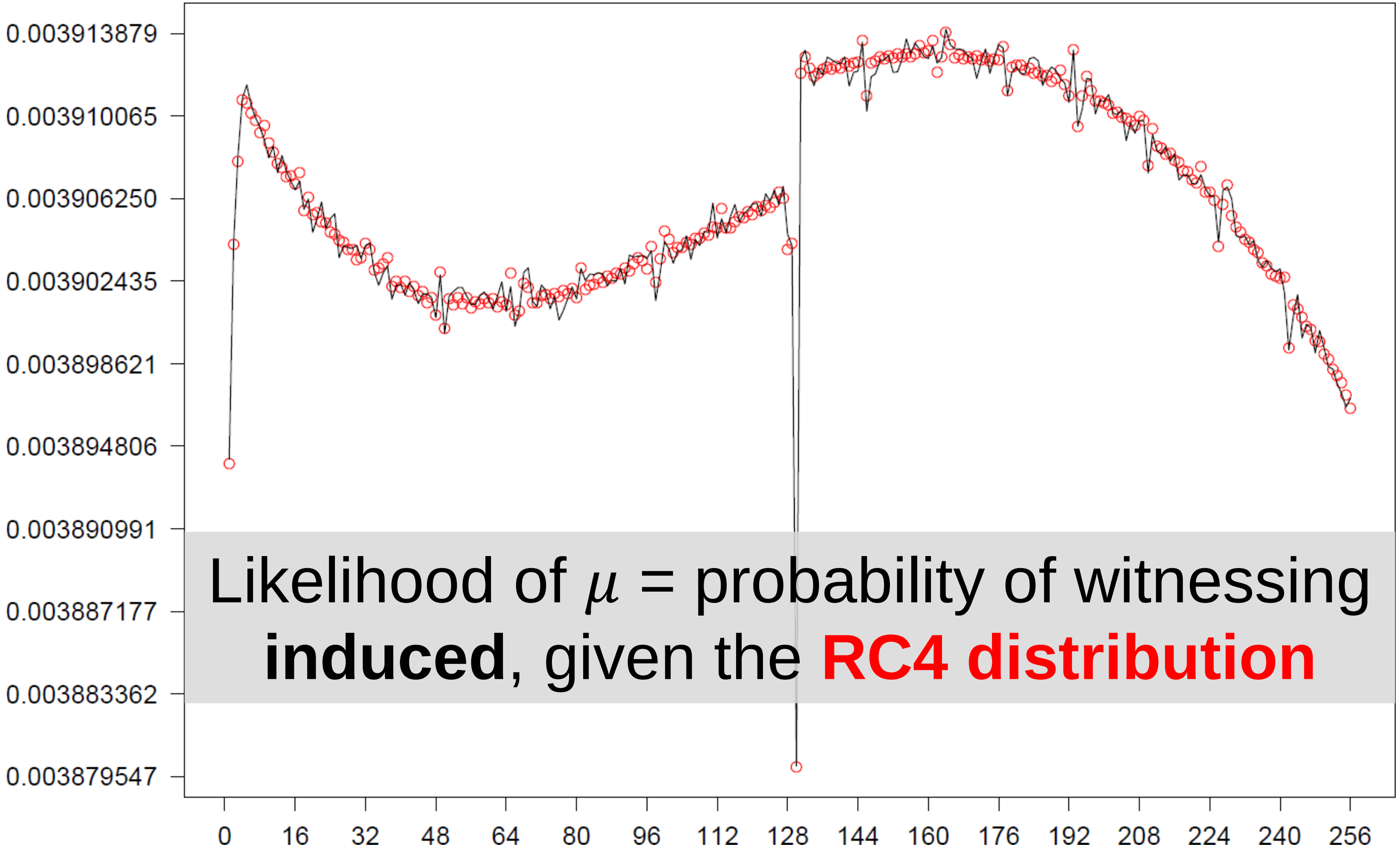
If plaintext byte $\mu = \mathtt{0x5C}$: **RC4** & **Induced**



Likelihood of μ = probability of witnessing **induced**, given the **RC4 distribution**

Example: Decrypt byte 1

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



Likelihood of μ = probability of witnessing **induced**, given the **RC4 distribution**