Recent Results on AE in WPA

Kenny Paterson, Bertram Poettering, Jacob Schuldt

Information Security Group

ROYAL HOLLOWAY UNIVERSITY OF LONDON

Overview

- Introduction to WPA/TKIP
- Biases in RC4 keystreams
- Biases in WPA/TKIP keystreams
- Plaintext recovery attack for the repeated plaintext setting
- (Advertising break)
- Exploiting TSCs for improved attacks
- Concluding remarks/open problems

Introduction to WPA/TKIP

- WEP, WPA, WPA2 are all IEEE standards for wireless LAN encryption under the 802.11 family.

- WEP (1999) is considered to be badly broken
  - Key recovery attacks based on Rc4 weaknesses and construction of RC4 key from concatenation of known 24-bit IV and unknown, but fixed key.
  - Beginning with Fluhrer, Mantin, Shamir, now roughly 10-20k packets needed for key recovery.
  - Other attacks on integrity, authentication.

- WPA/TKIP was proposed by IEEE in 2003 as an intermediate solution.
  - Allow reuse of same hardware, firmware-only upgrade.
  - Hence only limited changes to WEP design were possible.
  - Introduction of supposedly better per-frame keys (TKIP: Temporal Key Integrity Protocol).
  - Introduction of MIC integrity algorithm to prevent active traffic modification attacks.

Introduction to WPA/TKIP

- WPA was only intended as a temporary fix.
- WPA2 (2004) introduces a strong cryptographic solution based on AES-CCM.
  - Also includes optional support for TKIP.

- But WPA is still in widespread use today.
- Vanhoef-Piessens (2013): 71% of 6803 networks surveyed still permit WPA/TKIP; 19% allowed ONLY WPA/TKIP.

- This makes the continued analysis of the security of WPA/TKIP a worthwhile activity.

Previous attacks on WPA/TKIP

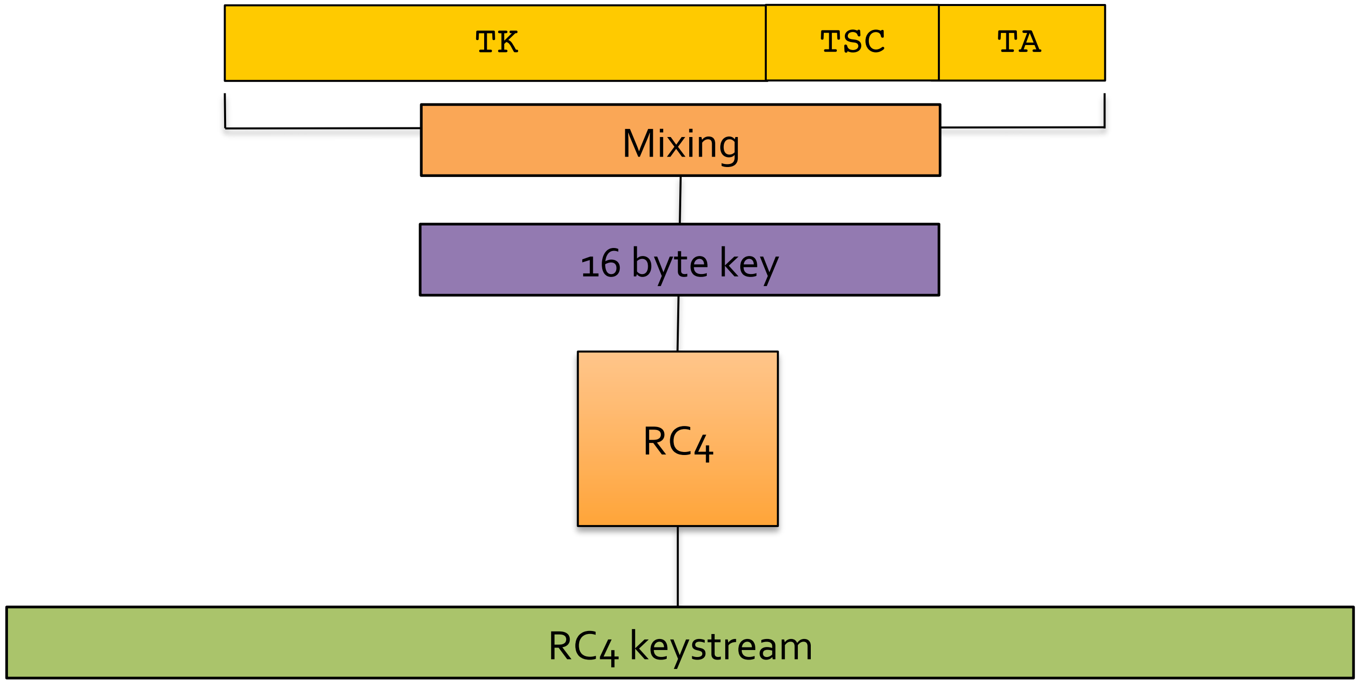
- Previous attacks were active and slow, or required large amounts of known plaintext and computation.

- Tews-Beck (2009):
  - Rate-limited plaintext recovery.
  - Active attack based on chop-chop method for recovering plaintext bytes one-by-one.
  - Requires support for alternative QoS channels to bypass WPA's anti-replay protection.
  - Rate-limited because correctness of plaintext guess indicated by MIC verification failure, and only 2 MIC failures per minute are tolerated.

- Sepehrdad-Vaudenay-Vuagnoux (2011):
  - Statistical key recovery attack using known plaintexts from 2^{38} frames and 2^{96} computation.

Overview of WPA/TKIP encryption

- `TK` (Temporal Key): 128 bits, used to protect many consecutive frames.
- `TSC` (TKIP Sequence Counter) : 48 bits, incremented for each frame sent, sent in frames.
- `TA` (Transmitter Address): 48 bits, MAC address of sender, sent in frames.



RC4

RC4 State

Byte permutation $\mathcal{S}$ and indices i and j

RC4 Key scheduling

```
begin
    for $i = 0$ to 255 do
        $\mathcal{S}[i] \leftarrow i$
    end
    $j \leftarrow 0$
    for $i = 0$ to 255 do
        $j \leftarrow j + \mathcal{S}[i] + K[i \bmod keylen] \bmod 256$
        swap($\mathcal{S}[i], \mathcal{S}[j]$)
    end
    $i, j \leftarrow 0$
end
```

RC4 Keystream generation

```
begin
    $i \leftarrow i + 1 \bmod 256$
    $j \leftarrow j + \mathcal{S}[i] \bmod 256$
    swap($\mathcal{S}[i], \mathcal{S}[j]$)
    $Z \leftarrow \mathcal{S}[\ \mathcal{S}[i] + \mathcal{S}[j] \bmod 256\ ]$
    return $Z$
end
```

Biases in RC4 keystreams

Single-byte biases in the RC4 keystream

Z_i = value of i-th keystream byte

[Mantin-Shamir 2001]:

$$\Pr[Z_2 = 0] \approx \frac{1}{128}$$

[Mironov 2002]:

Described distribution of Z_1 (bias away from 0, sine-like distribution)

[Maitra-Paul-Sen Gupta 2011]: for $3 \leq r \leq 255$

$$\Pr[Z_r = 0] = \frac{1}{256} + \frac{c_r}{256^2} \qquad 0.242811 \leq c_r \leq 1.337057$$

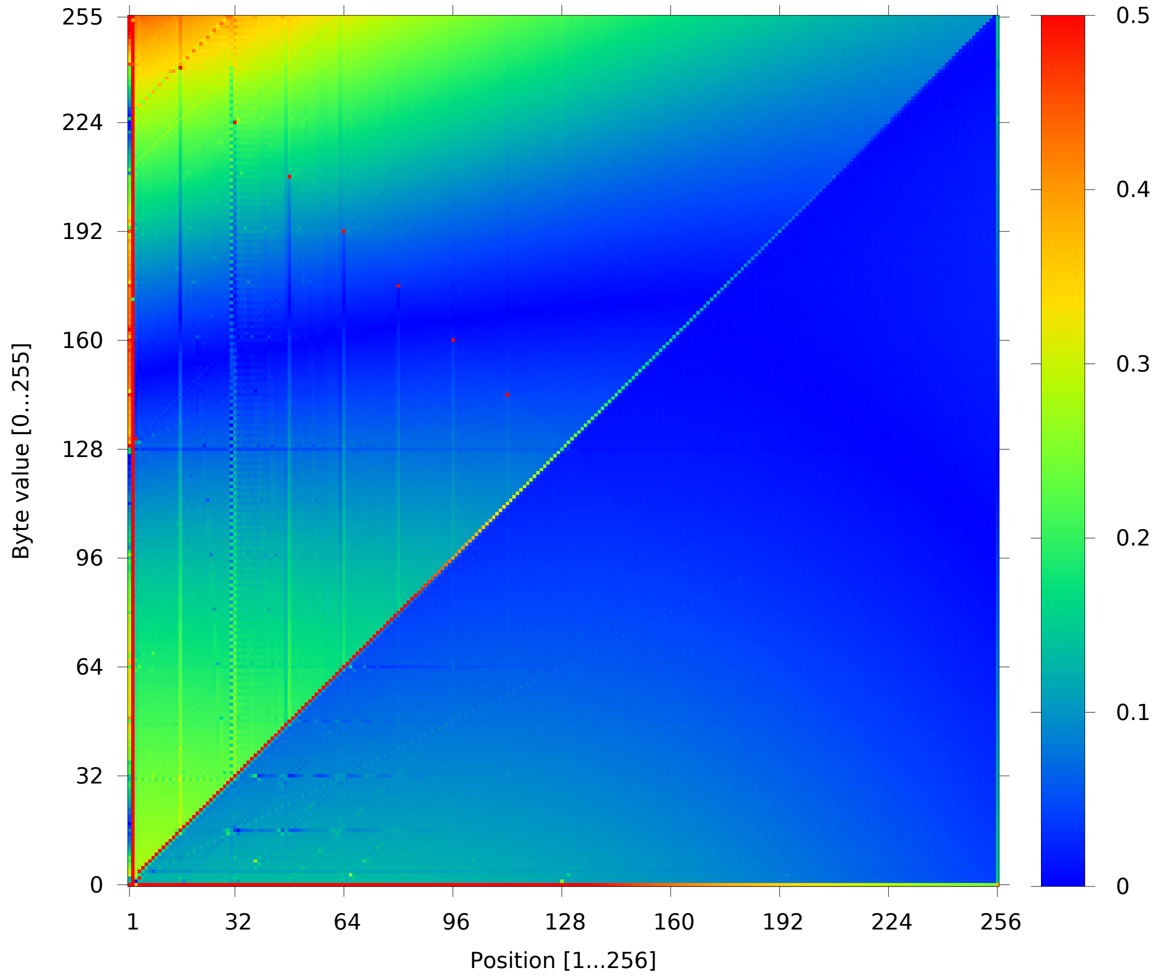
[Sen Gupta-Maitra-Paul-Sarkar 2011]:

$$\Pr[Z_l = 256 - l] \geq \frac{1}{256} + \frac{1}{256^2}$$ l = keylength

Alternative approach

- Compute!
- AlFardan-Bernstein-P.-Poettering-Schuldt (2013) considered the RC4 keystream distributions arising from 2^{45} random 128-bit keys…

All the RC4 biases (random 128-bit Keys)



x-axis: position; y-axis: keystream byte value; colour encodes bias size.

RC4 with random 128-bit keys

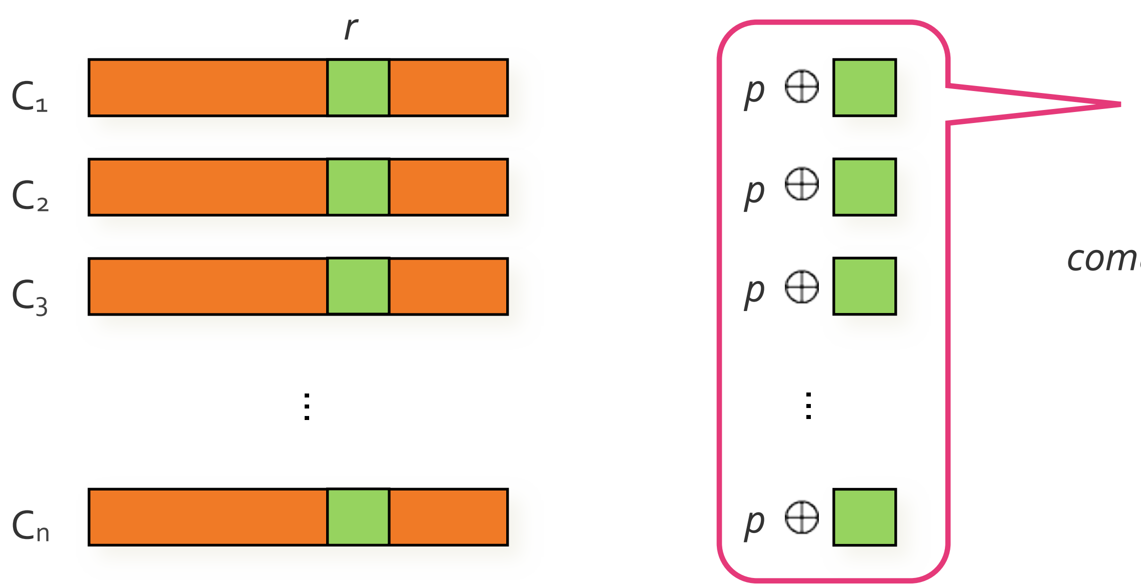
- Rc4 with random 128-bit keys has additional significant biases in *all* of its initial keystream bytes.

- Such biases enable recovery of plaintext in relevant keystream positions if sufficiently many encryptions of the same plaintext are available.
  - Using simple Bayesian statistical analysis.
  - Can be formally justified using hypothesis testing and log likelihood ratios.
  - *Multi-session* or *broadcast attack* scenario.

Plaintext recovery using keystream biases

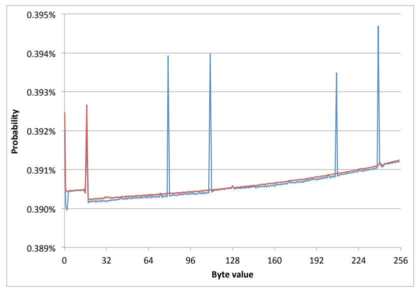
Encryptions of fixed plaintext under different keys

Plaintext candidate byte *p*



yields induced distribution on keystream byte *Z*

combine with known distribution



$\Downarrow$

Likelihood of *p* being correct plaintext byte

<u>Recovery algorithm:</u>
Compute most likely plaintext byte

Details of statistical analysis

Let $\boldsymbol{c}$ be the n-vector of ciphertext bytes in position r.

Let $\boldsymbol{q} = (q_{00}, q_{01}, \ldots, q_{ff})$ be the vector of keystream byte probabilities in position r.

Bayes theorem:

$$\begin{aligned}\Pr[P=p \mid \boldsymbol{C}=\boldsymbol{c}] &= \Pr[\boldsymbol{C}=\boldsymbol{c} \mid P=p].\Pr[P=p]/\Pr[\boldsymbol{C}=\boldsymbol{c}] \\ &= \Pr[\boldsymbol{Z}=\boldsymbol{c} \oplus p \mid P=p].\Pr[P=p]/\Pr[\boldsymbol{C}=\boldsymbol{c}].\end{aligned}$$

Assume $\Pr[P=p]$ is constant; $\Pr[\boldsymbol{C}=\boldsymbol{c}]$ is independent of the choice of p.

Then to maximise $\Pr[P=p \mid \boldsymbol{C}=\boldsymbol{c}]$ over all choices of p, we simply need to maximise

$$\Pr[\boldsymbol{Z}=\boldsymbol{c} \oplus p \mid P=p] = q_{00}^{n_{00}} q_{01}^{n_{01}} \ldots q_{ff}^{n_{ff}}$$

where n_x is the number of occurrences of byte value x in $\boldsymbol{Z}=\boldsymbol{c} \oplus p$.

Work with logs to simplify computations; calculate $\boldsymbol{q}$ empirically by sampling.

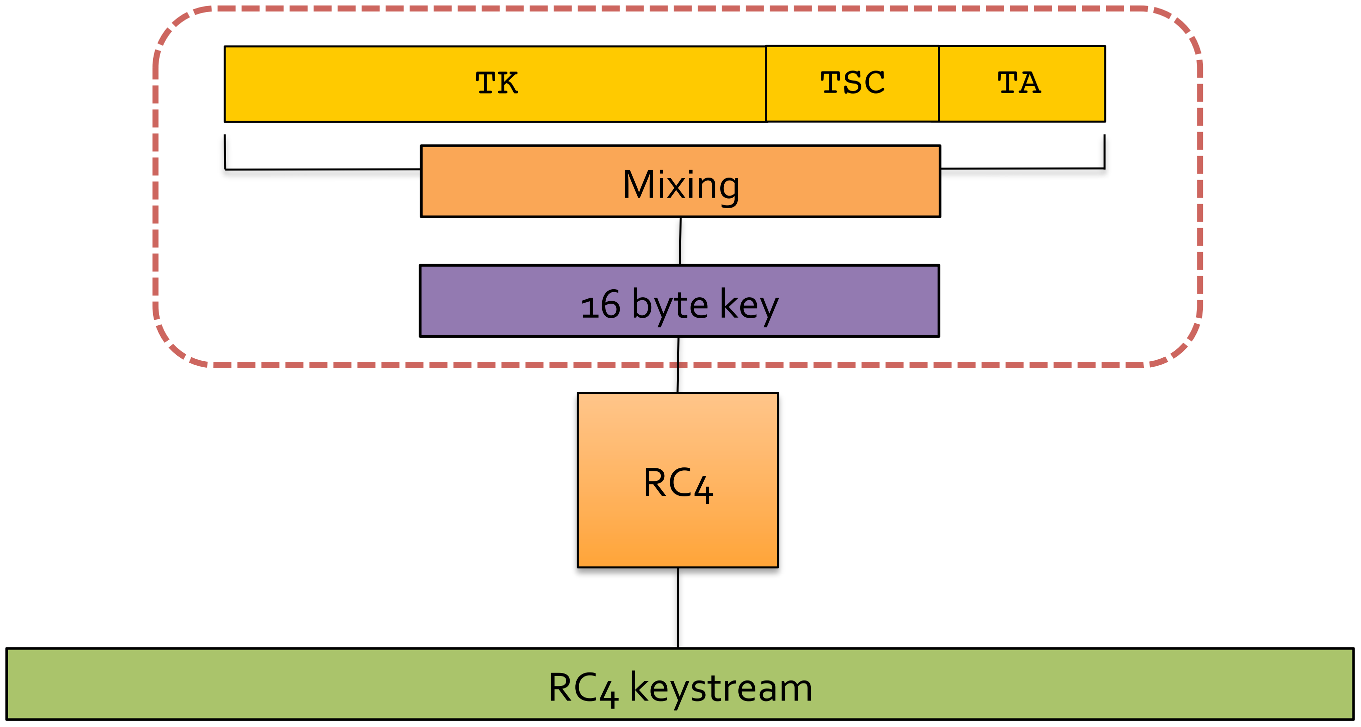
Applications

- Technique successfully applied to Rc4 as used in SSL/TLS by AlFardan-Bernstein-P.-Poettering-Schuldt (2013).
  - 50% of all SSL/TLS traffic (was) protected using RC4!
  - Attack realisable in TLS context using client-side Javascript, resulting in recovery of session cookies.
  - (Preferred version of attack actually exploits Fluhrer-McGrew double-byte biases.)

- So what about Rc4 with WPA/TKIP keys?
  - Every frame transmitted on the network is encrypted with a new key, so natural repeated plaintext scenario.
  - Candidates for repeated plaintext bytes: fixed but unknown fields in protocol headers; Javascript-in-the-browser attack on HTTP traffic also still possible.
  - But WPA/TKIP does not use random 128-bit keys….

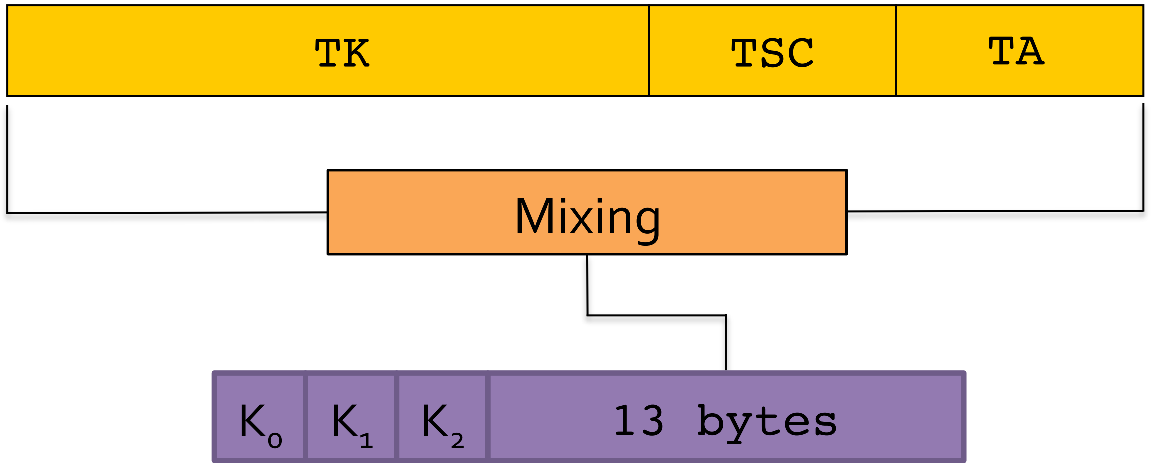
Overview of WPA/TKIP encryption

- `TK` (Temporal Key): 128 bits, used to protect many consecutive frames.
- `TSC` (TKIP Sequence Counter) : 48 bits, incremented for each frame sent.
- `TA` (Transmitter Address): 48 bits, MAC address of sender.



WPA/TKIP key mixing function

16-byte RC4 keys in WPA/TKIP are derived from `TK`, `TSC` and `TA` using a key mixing function



$K_0 = TSC_1$
$K_1 = (TSC_1 \text{ OR } 0x20) \text{ AND } 0x7f$
$K_2 = TSC_0$

(TSC_0 and TSC_1 are the two least significant bytes of `TSC`)

Biases in WPA/TKIP keystreams

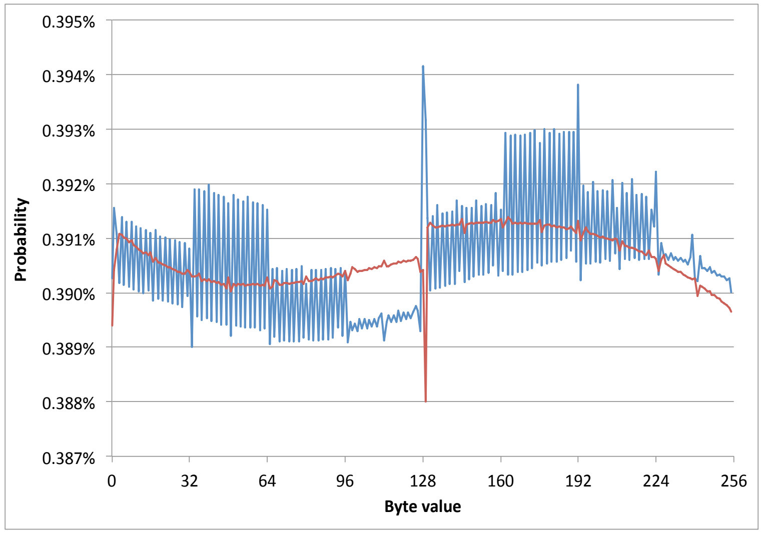
Biases in WPA/TKIP keystreams

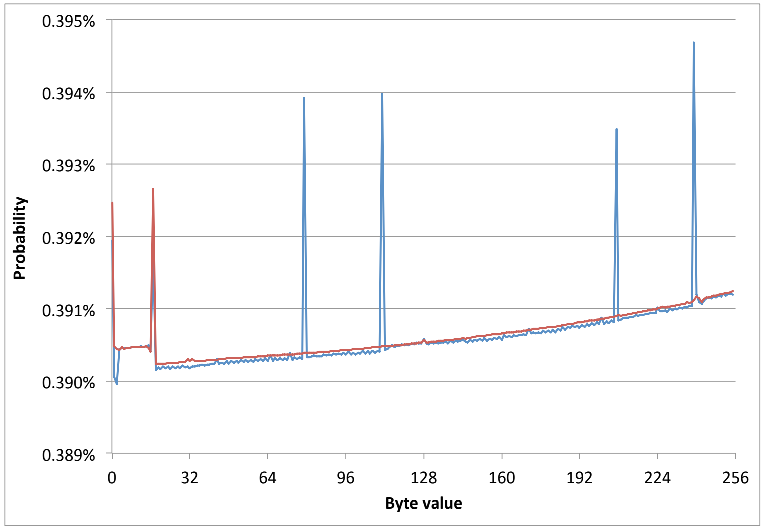
- WPA/TKIP keys have additional structure:

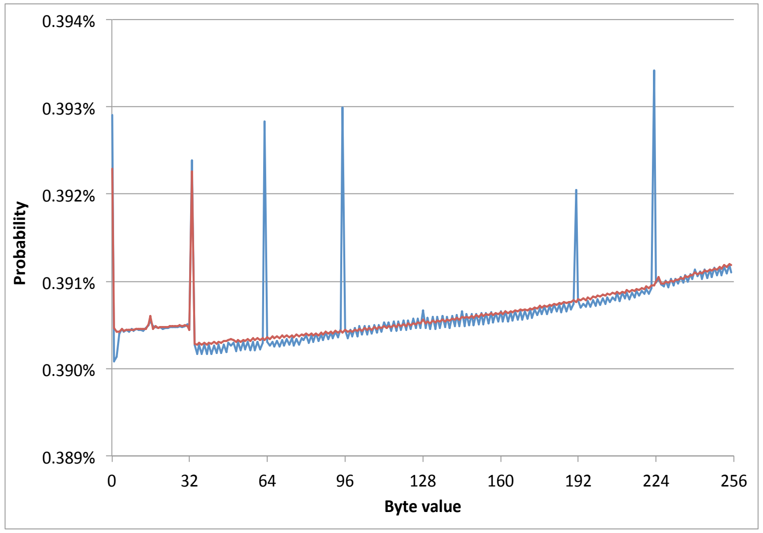
```
K_0 =  TSC_1
K_1 = (TSC_1 OR 0x20) AND 0x7f
K_2 =  TSC_0
```

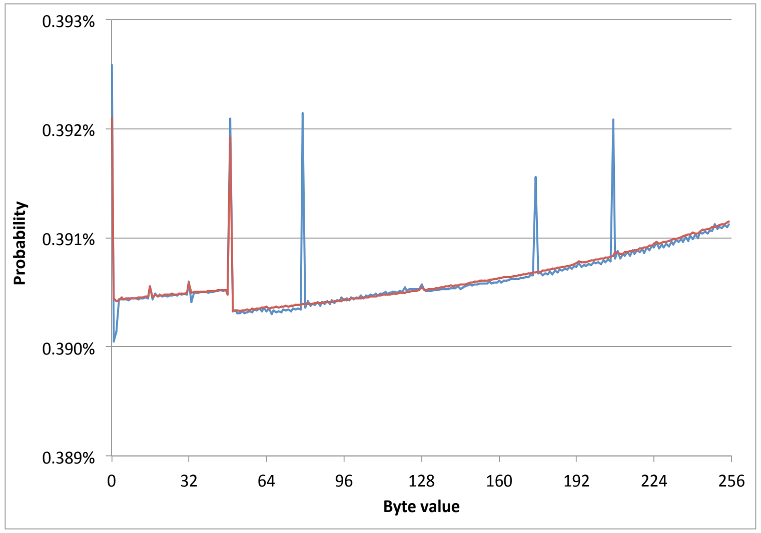
- Might this additional structure lead to more and/or bigger keystream biases?

RC4 with WPA/TKIP keys: keystream bytes 1, 17, 33, 49

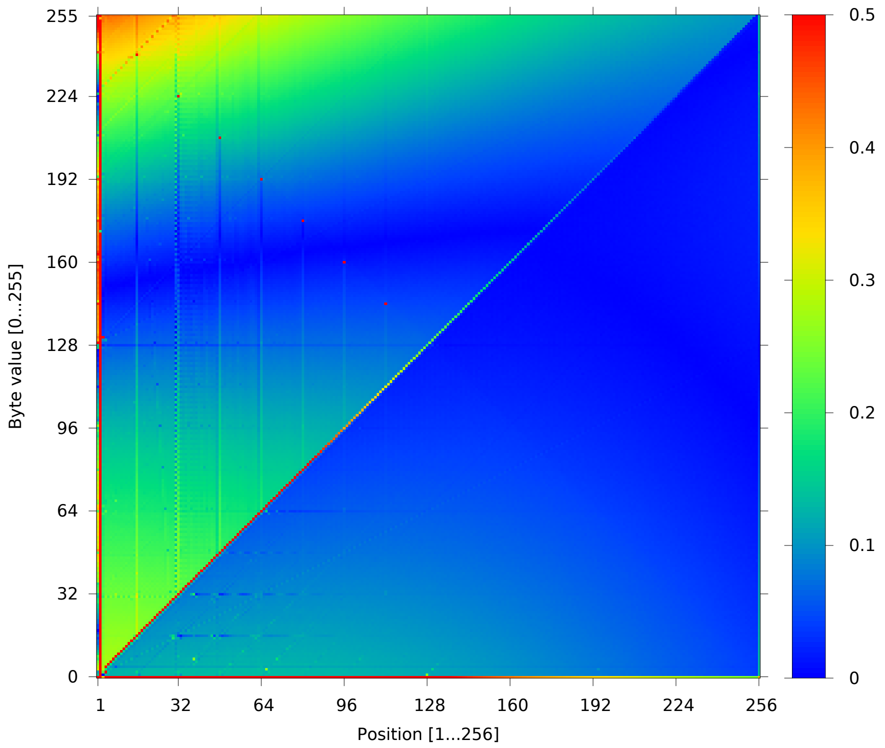




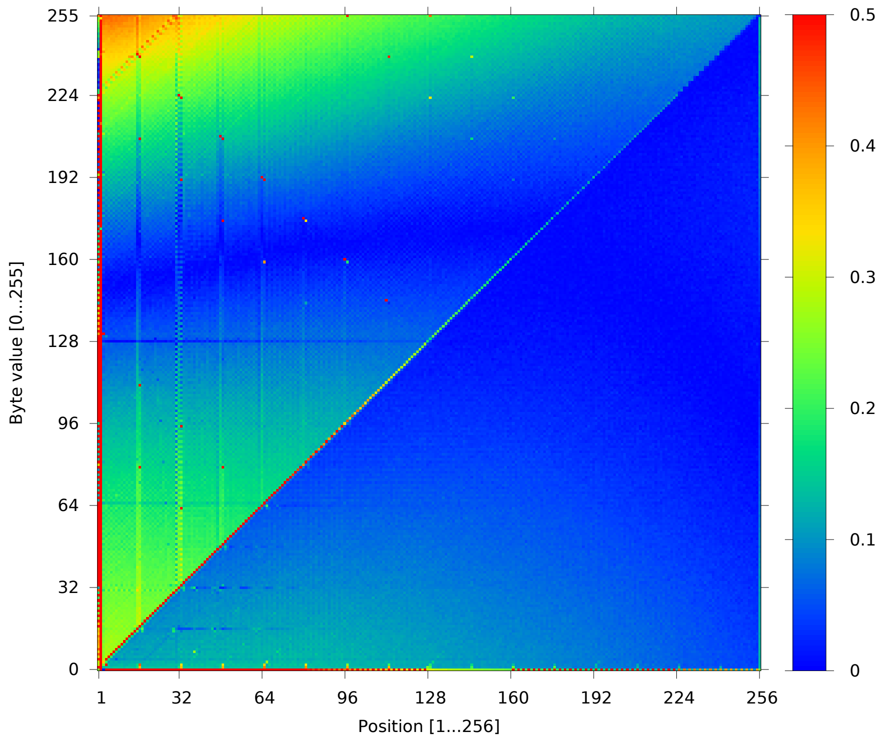




Comparing biases in RC4 with random 128-bit keys and WPA/TKIP keys



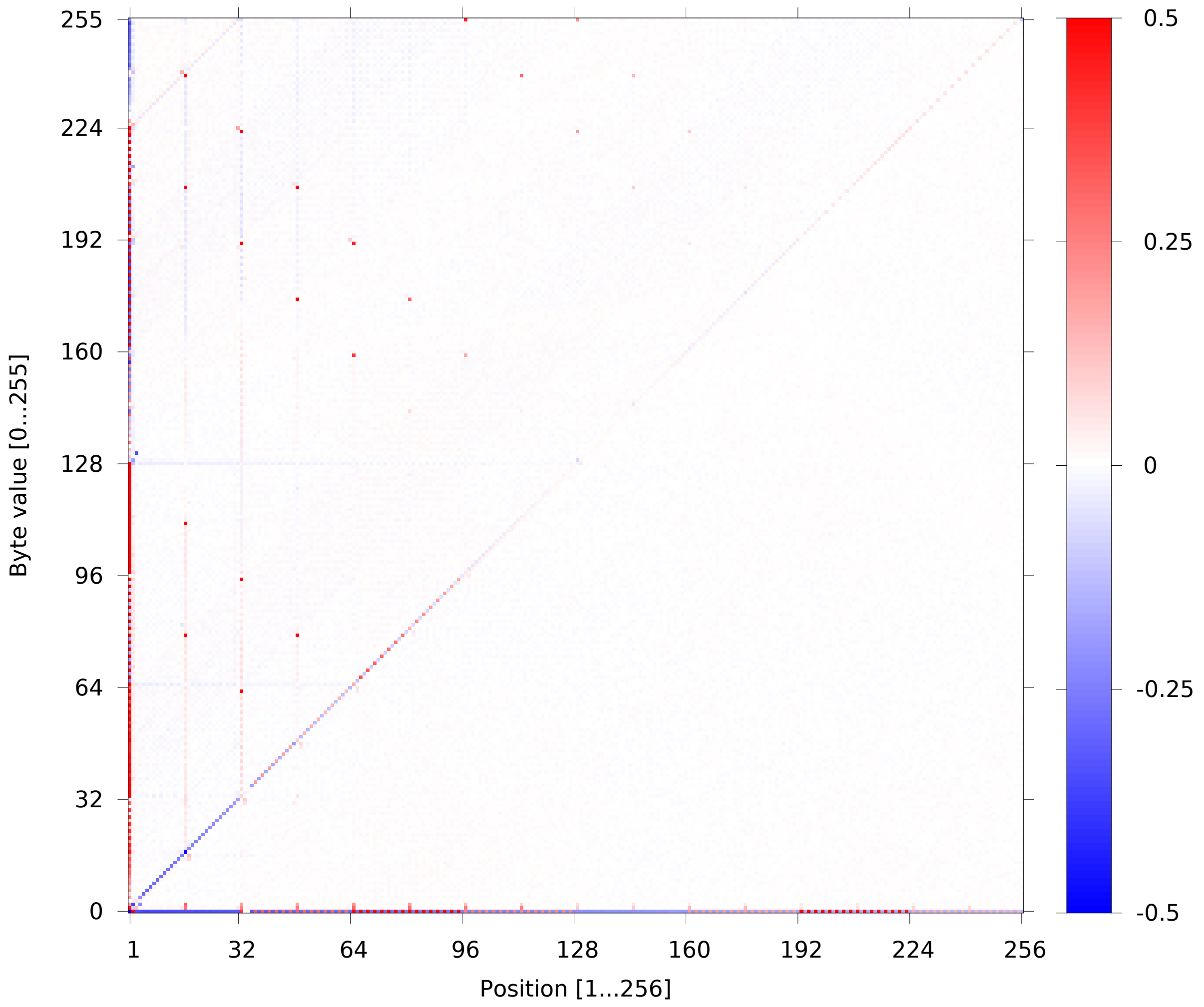
Random 128-bit keys.



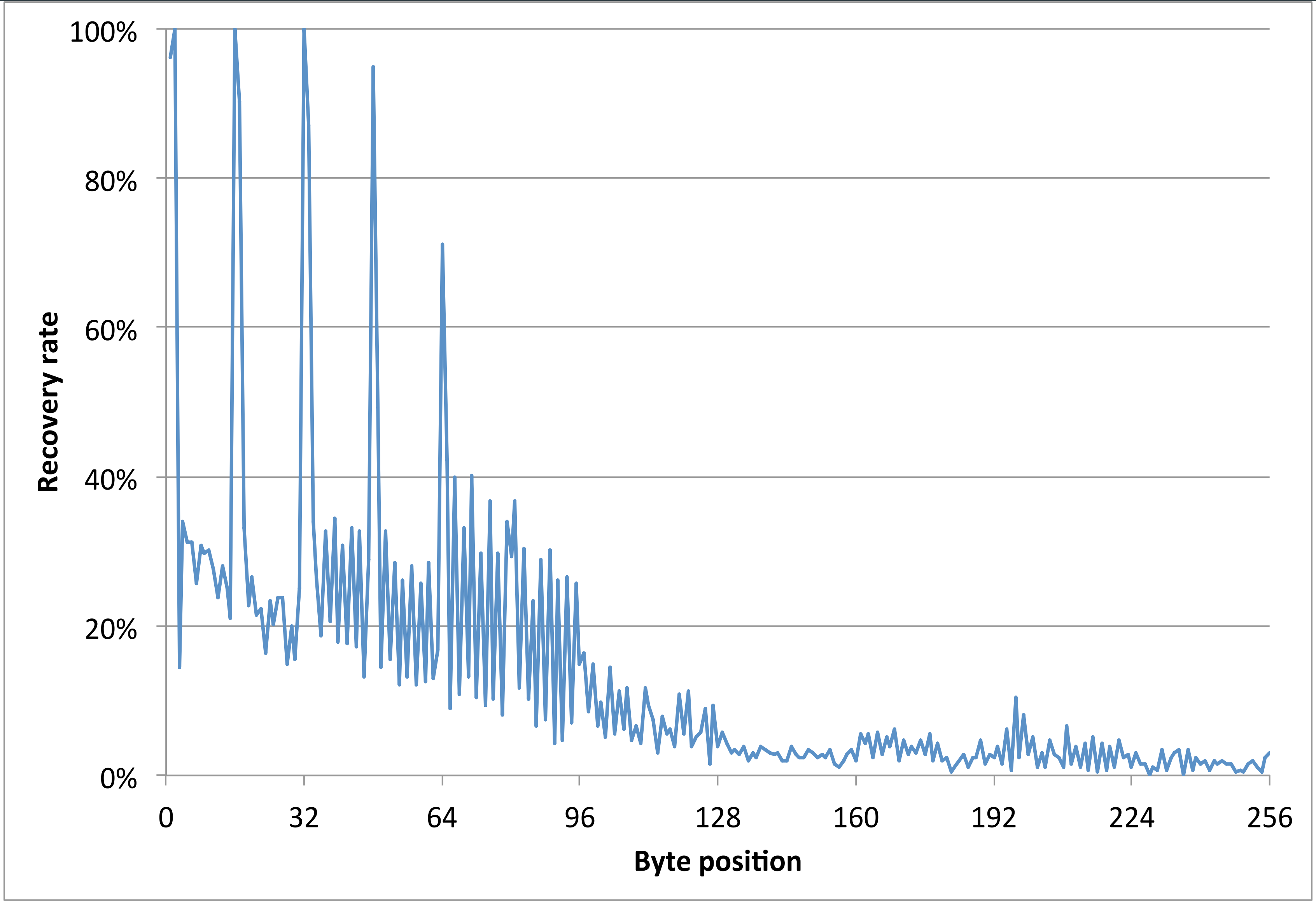
WPA/TKIP keys.

x-axis: position; y-axis: keystream byte value;
colour encodes bias size.

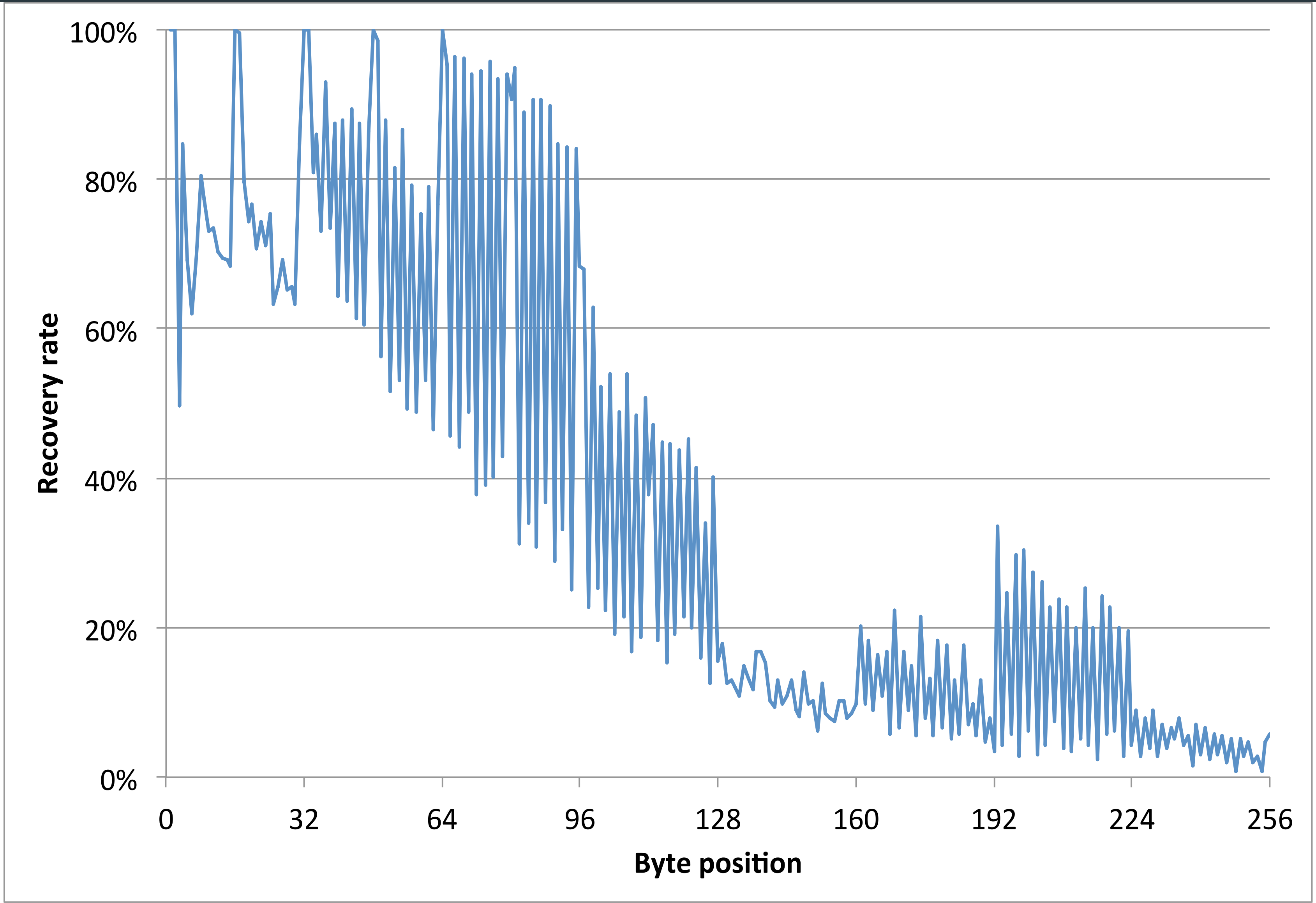
Comparing biases in RC4 with random 128-bit keys and WPA/TKIP keys



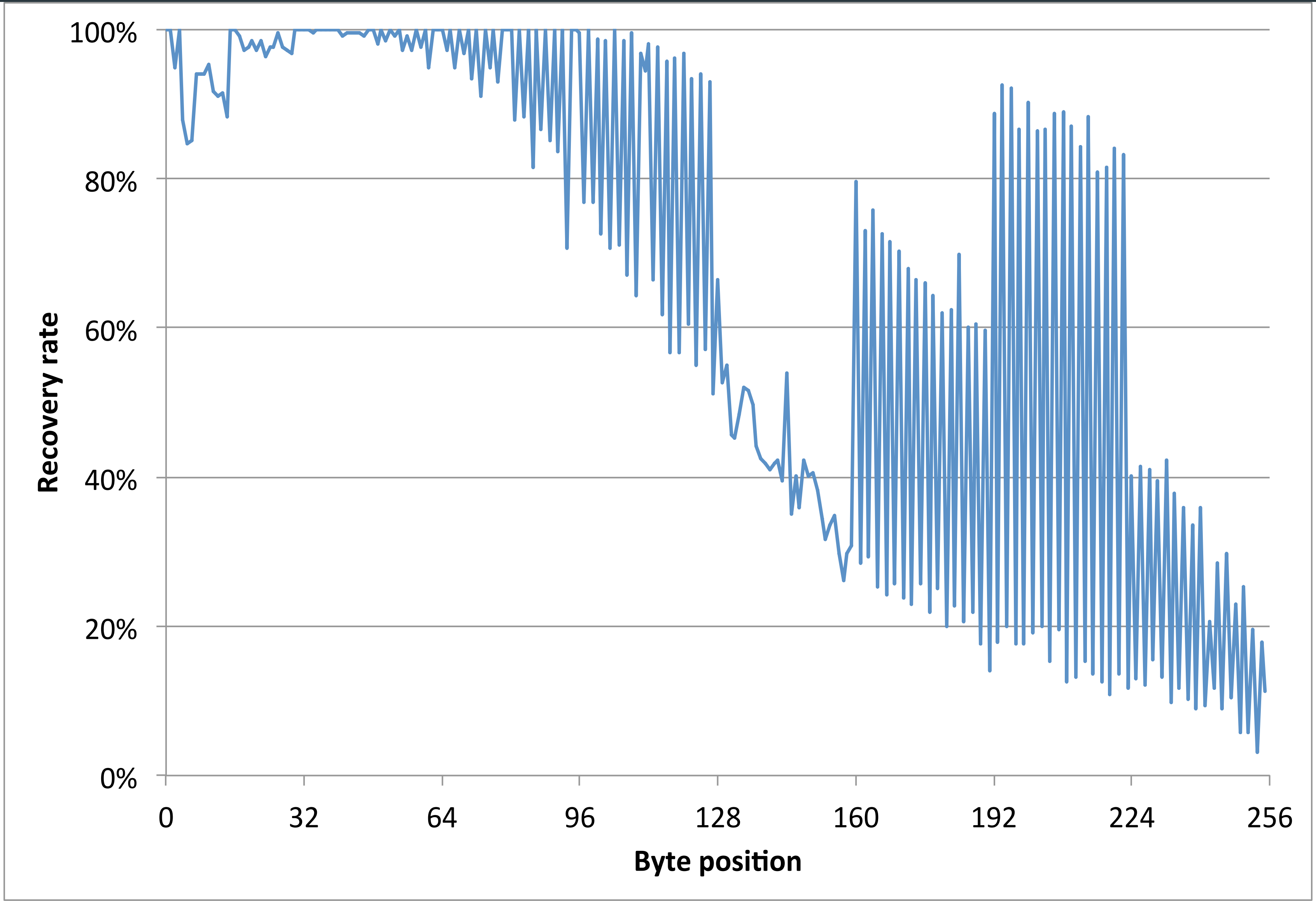
Plaintext recovery attack: 2^{24} frames



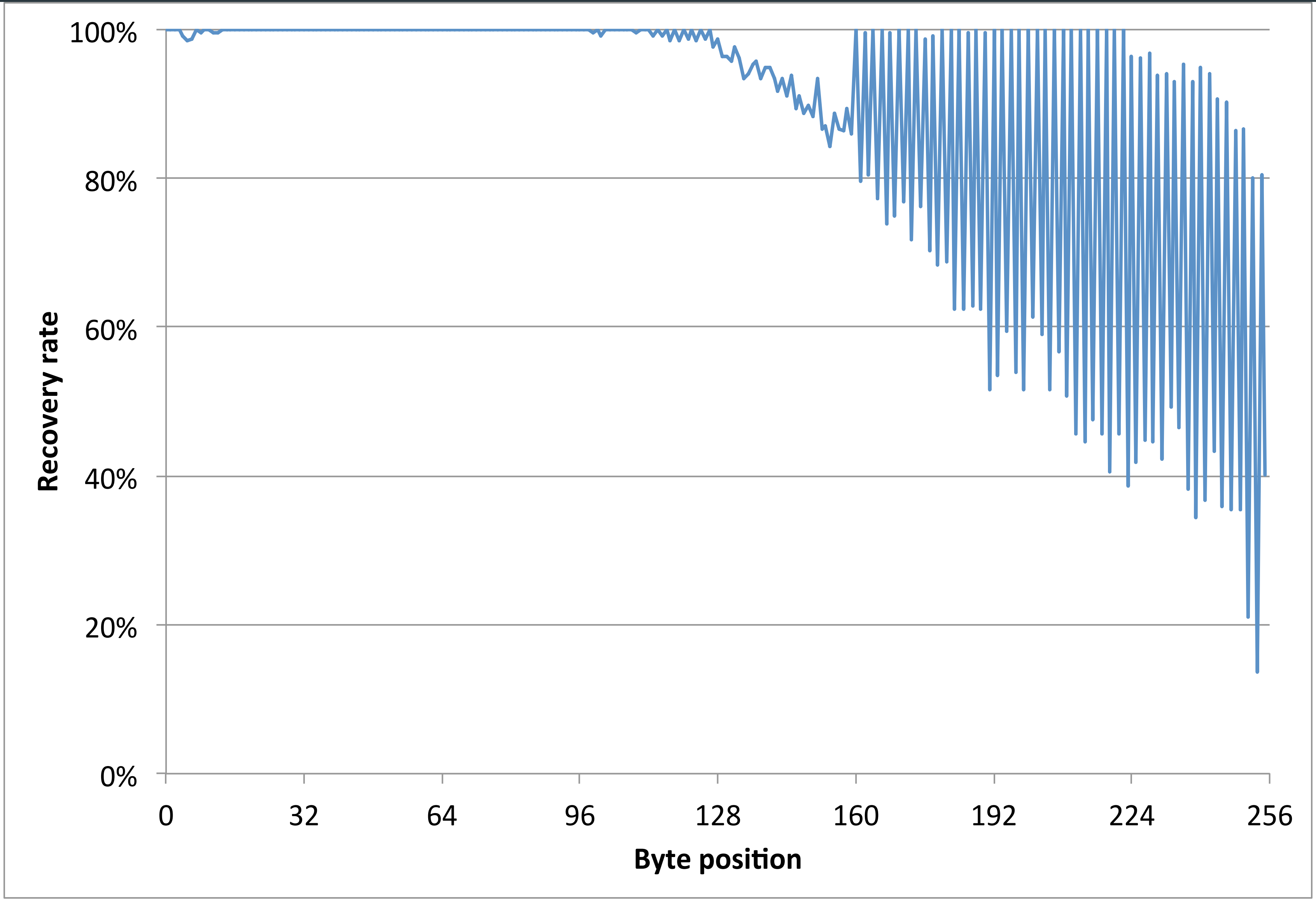
Plaintext recovery attack: 2^{26} frames



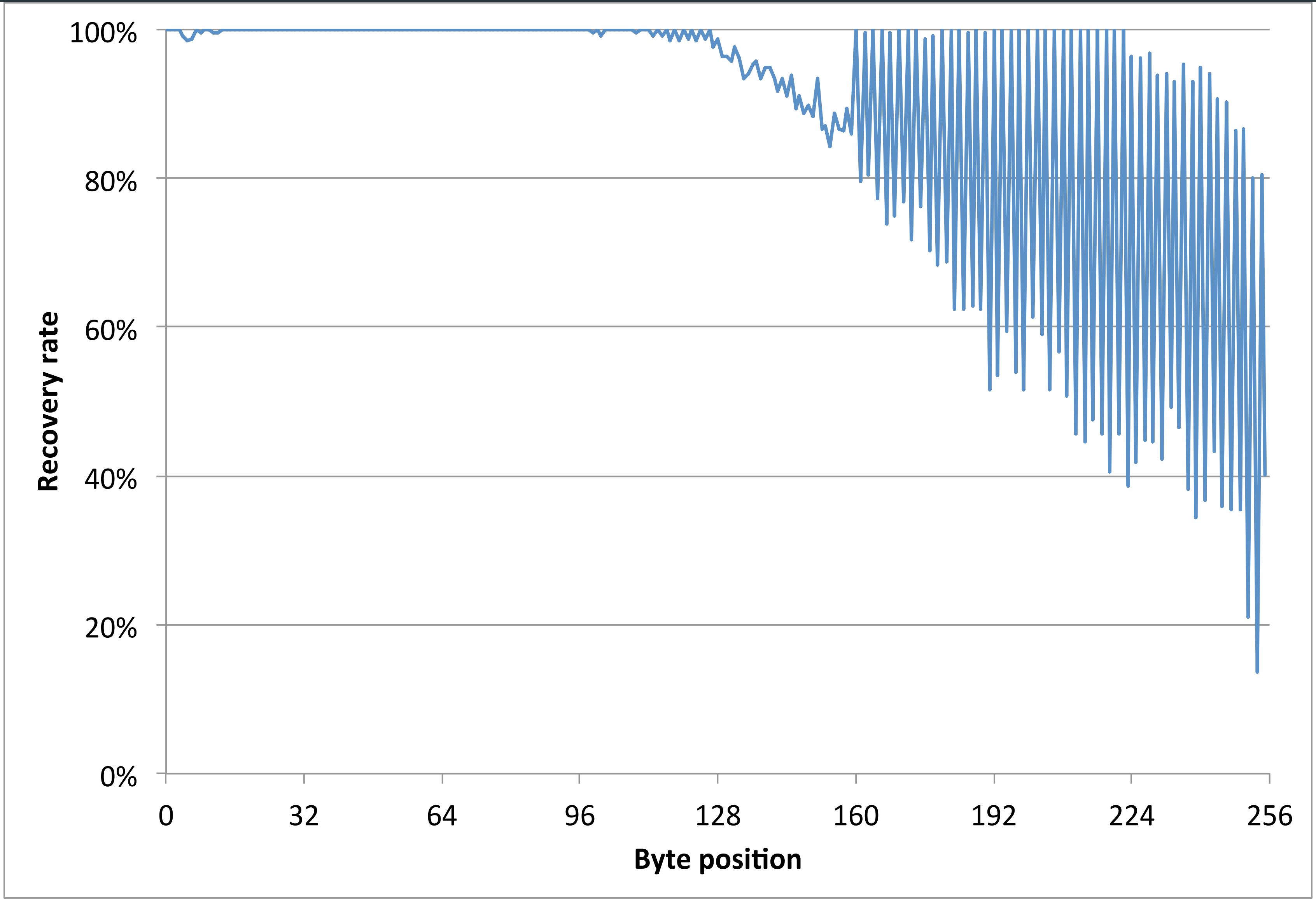
Plaintext recovery attack: 2^{28} frames



Plaintext recovery attack: 2^{30} frames



Plaintext recovery attack: 2^{30} frames



(Advertising break)

Real World Cryptography 2015

London, UK, 7-9 January 2015

http://www.realworldcrypto.com/rwc2015

#realworldcrypto

Speakers to include:

Elena Andreeva (K.U. Leuven)

Dan Bogdanov (Cybernetica)

Sasha Boldyreva (Georgia Tech)

Claudia Diaz (K.U. Leuven)

Roger Dingledine (Tor project)

Ian Goldberg (U. Waterloo)

Arvind Mani (LinkedIn)

Luther Martin (Voltage Security)

Elisabeth Oswald (U. Bristol)

Scott Renfro (Facebook)

Ahmad Sadeghi (TU Darmstadt)

Elaine Shi (UMD)

Brian Sniffen (Akamai)

Nick Sullivan (CloudFlare)

Exploiting TSCs

Exploiting TSC information

- Recall that WPA/TKIP keys have additional structure:

$$K_0 = TSC_1$$

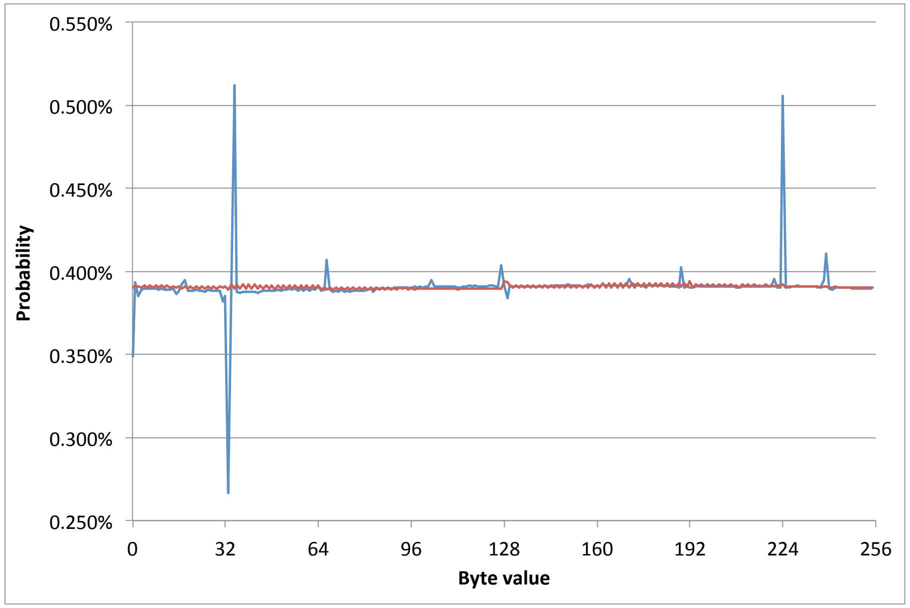
$$K_1 = (TSC_1 \text{ OR } 0x20) \text{ AND } 0x7f$$

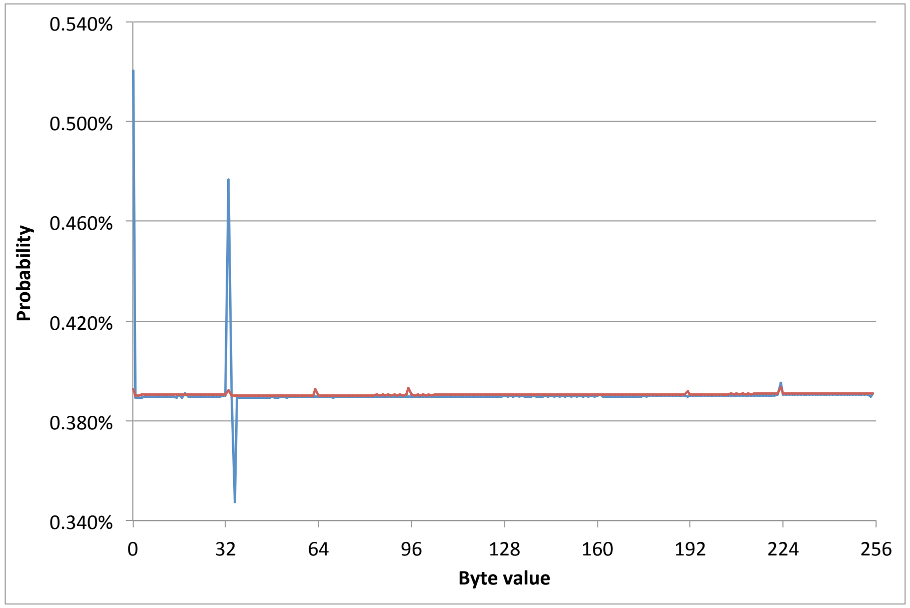
$$K_2 = TSC_0$$

- Recall also that the `TSC` is transmitted in clear as part of frame.
- **Idea**: there may be even larger keystream biases that arise for *specific* (TSC_0, TSC_1) values; these could disappear when aggregating over all (TSC_0, TSC_1) values.
- Exploitation in plaintext recovery attack:

1. bin available ciphertexts into 2^{16} bins according to (TSC_0, TSC_1) value;
2. carry out likelihood analysis in each bin using bin-specific keystream distribution;
3. combine likelihoods across bins to compute plaintext likelihoods.

Confirming existence of large ($\mathtt{TSC}_0$,$\mathtt{TSC}_1$) –specific biases





Output byte 1, ($\mathtt{TSC}_0$,$\mathtt{TSC}_1$) = (0x00,0x00)

Output byte 33, ($\mathtt{TSC}_0$,$\mathtt{TSC}_1$) = (0x00,0x00)

Blue: TSC-specific biases
Red: fully aggregated WPA/TKIP biases

Exploiting TSC information

- Problem: this approach requires a large number of keystreams to get accurate estimates for the $(\texttt{TSC}_0,\texttt{TSC}_1)$-specific keystream distributions.
- How large?
- At a minimum, we would like 2^{32} keystreams for each $(\texttt{TSC}_0,\texttt{TSC}_1)$ value, 2^{48} in all.
  - Fewer than 2^{32} gives noisy keystream estimates, adversely affecting plaintext recovery rate.
  - Ideally, we would have 2^{40} keystreams for each $(\texttt{TSC}_0,\texttt{TSC}_1)$ value, 2^{56} in all.
- With our local computing setup, computing 2^{24} keystreams for each of 2^{16} $(\texttt{TSC}_0,\texttt{TSC}_1)$ values required 2^6 core days of computation.
- Desired computation would then need 2^{14} core days (and ideally 2^{22} core days for very accurate keystream estimates).

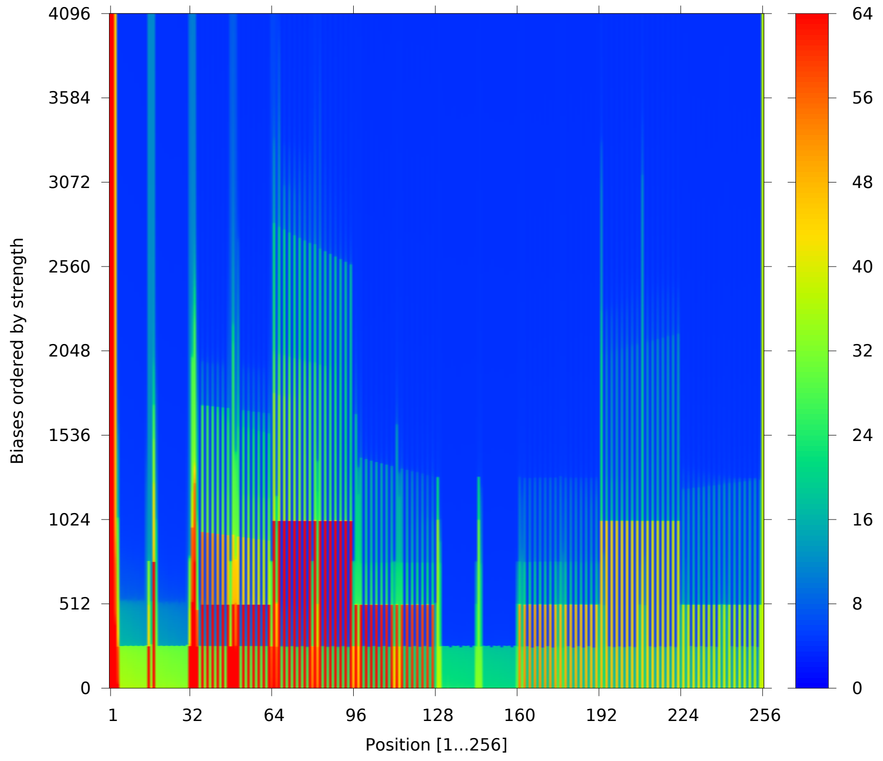
TSC_0 aggregation

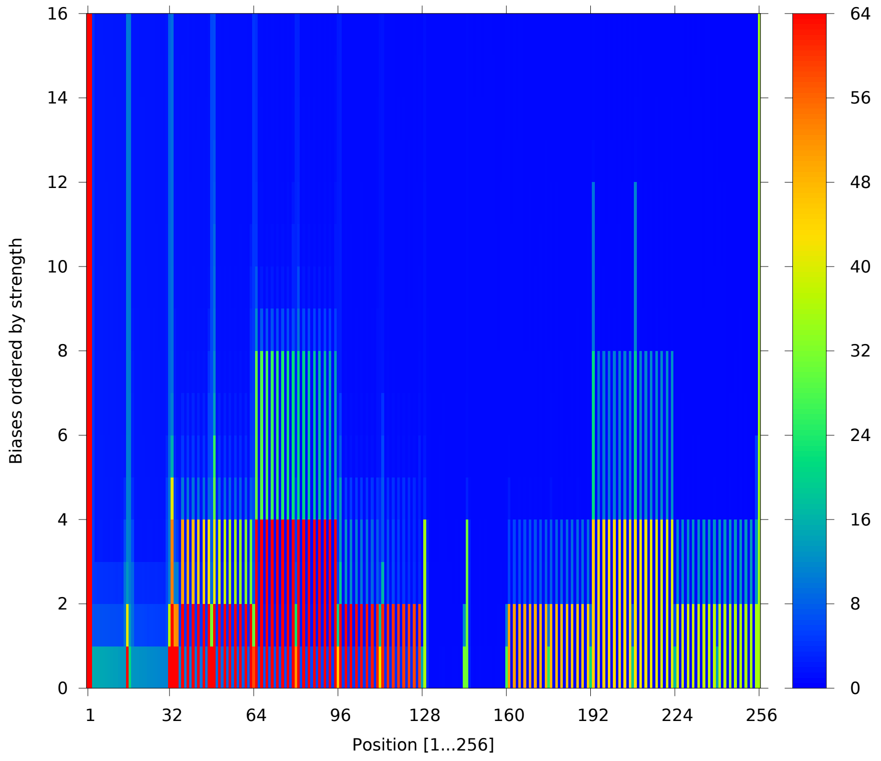
- In view of the computational challenges, another approach is needed…
- TSC_1 is used in computing two key bytes; TSC_0 in only one:

$$K_0 = TSC_1$$
$$K_1 = (TSC_1 \text{ OR } 0x20) \text{ AND } 0x7f$$
$$K_2 = TSC_0$$

- Hence we may expect biases to depend more strongly on TSC_1 than on TSC_0.
- Experiments bear this out.
- So we could ignore TSC_0 and look only at how biases depend on TSC_1.
- Effectively, we will aggregate biases over TSC_0, using 2^8 bins instead of 2^{16}.
- Our first attack can then be seen as the variant where we aggregate over *both* TSC_0 and TSC_1.

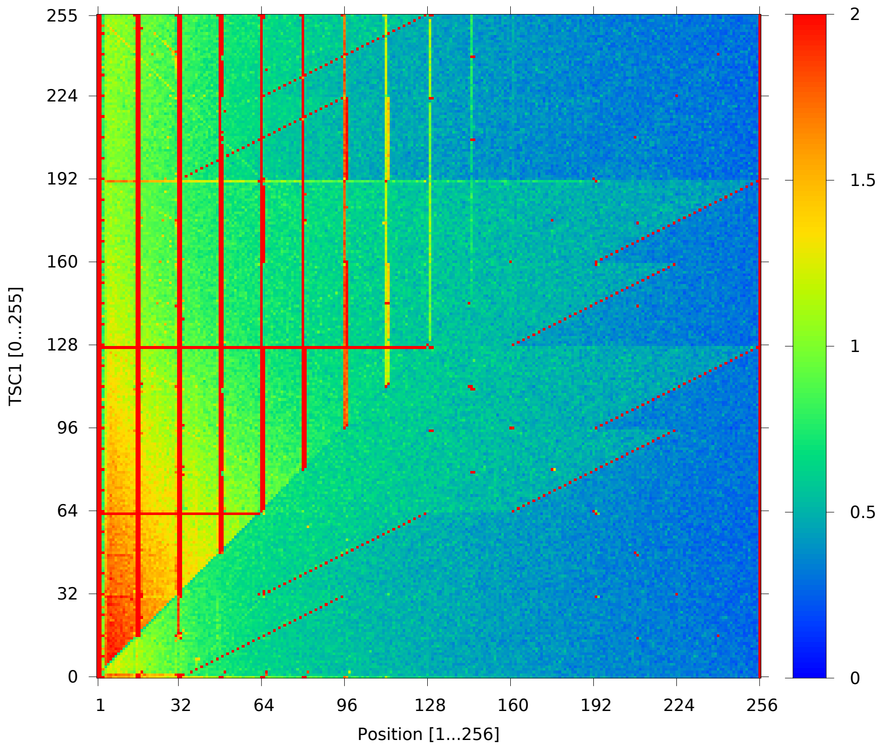
Effect of $\mathtt{TSC}_0$ aggregation



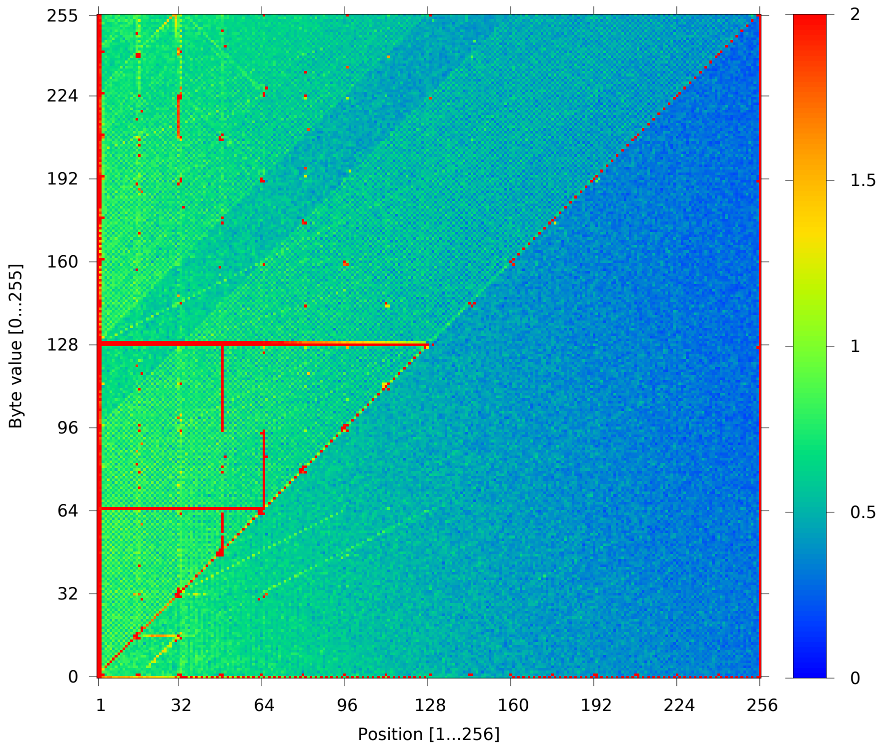


Numbers of large biases in TKIP keystream distributions for different keystream positions.
For each position, we show the strengths of the largest biases, sorted in descending order. Colouring scheme shows scaled bias size.
Left figure: all ($\mathtt{TSC}_0$,$\mathtt{TSC}_1$) pairs; right figure: $\mathtt{TSC}_0$-aggregated biases.

Locations of large biases ($\mathtt{TSC}_0$-aggregated)



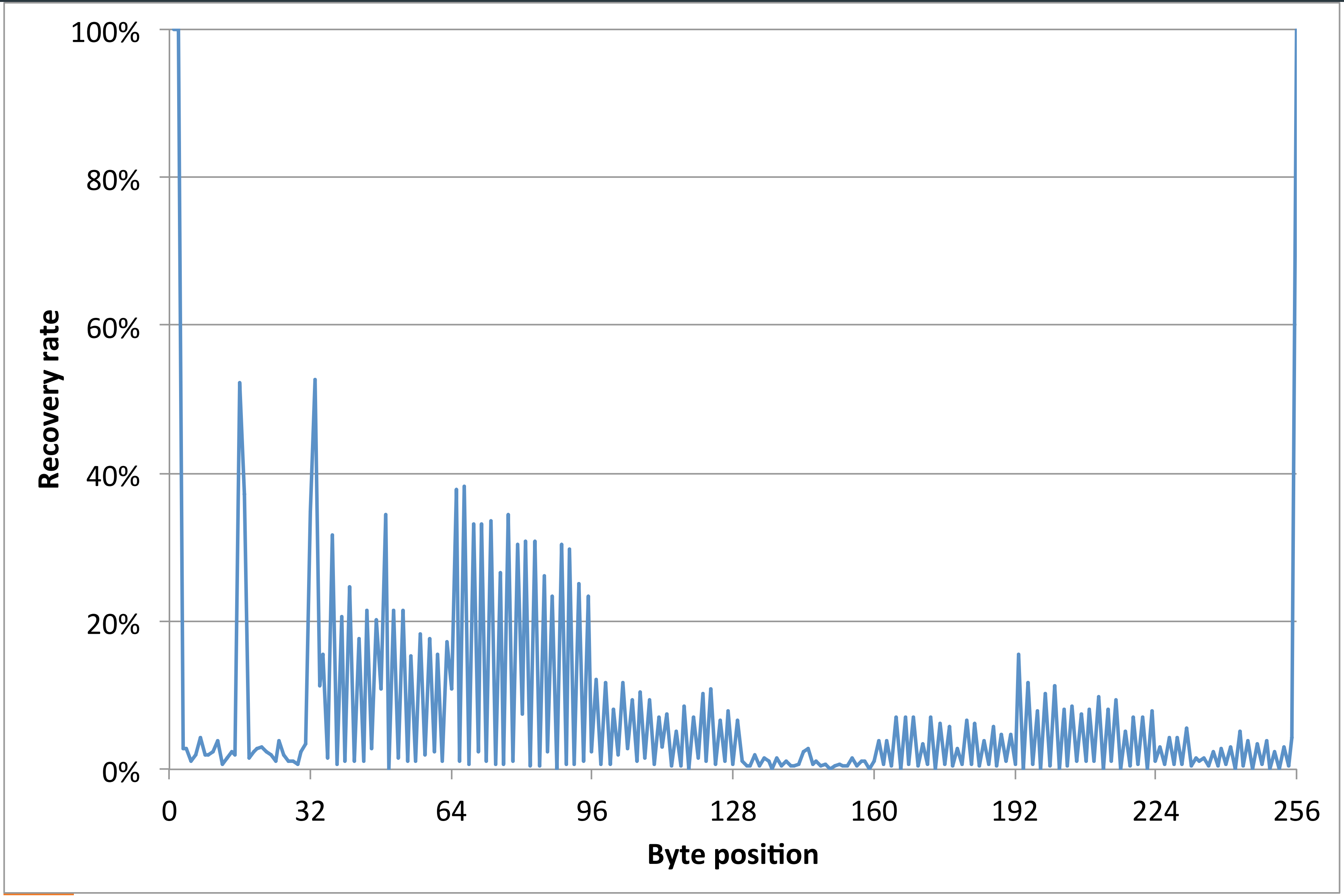
Strength of largest bias for each combination of position and $\mathtt{TSC}_1$ value.



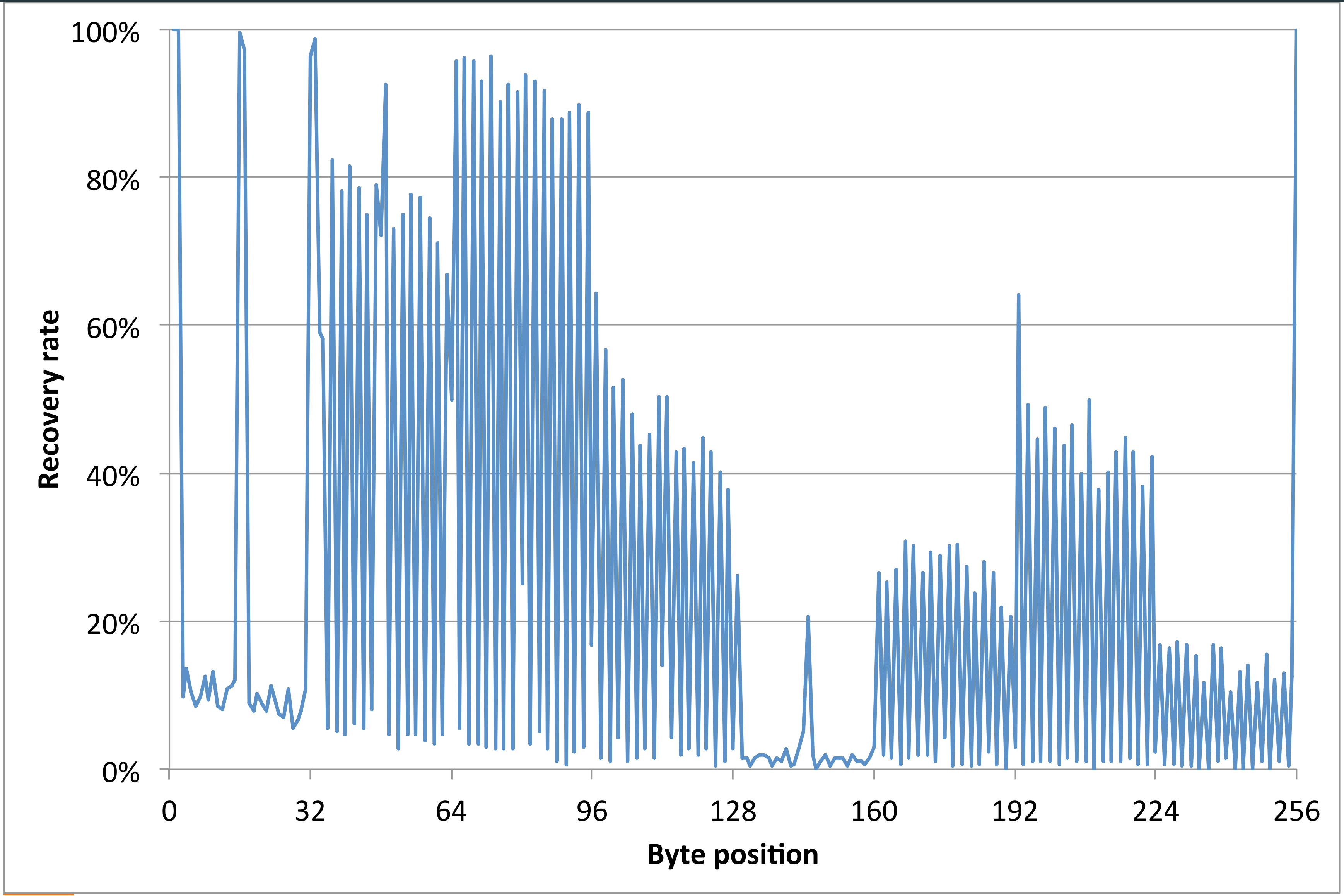
Strength of largest bias for each combination of position and byte value.

(Colouring scheme encodes the absolute difference between the biases and the (expected) probability 1/256, scaled up by a factor of 2^{16}, capped to a maximum of 2.)

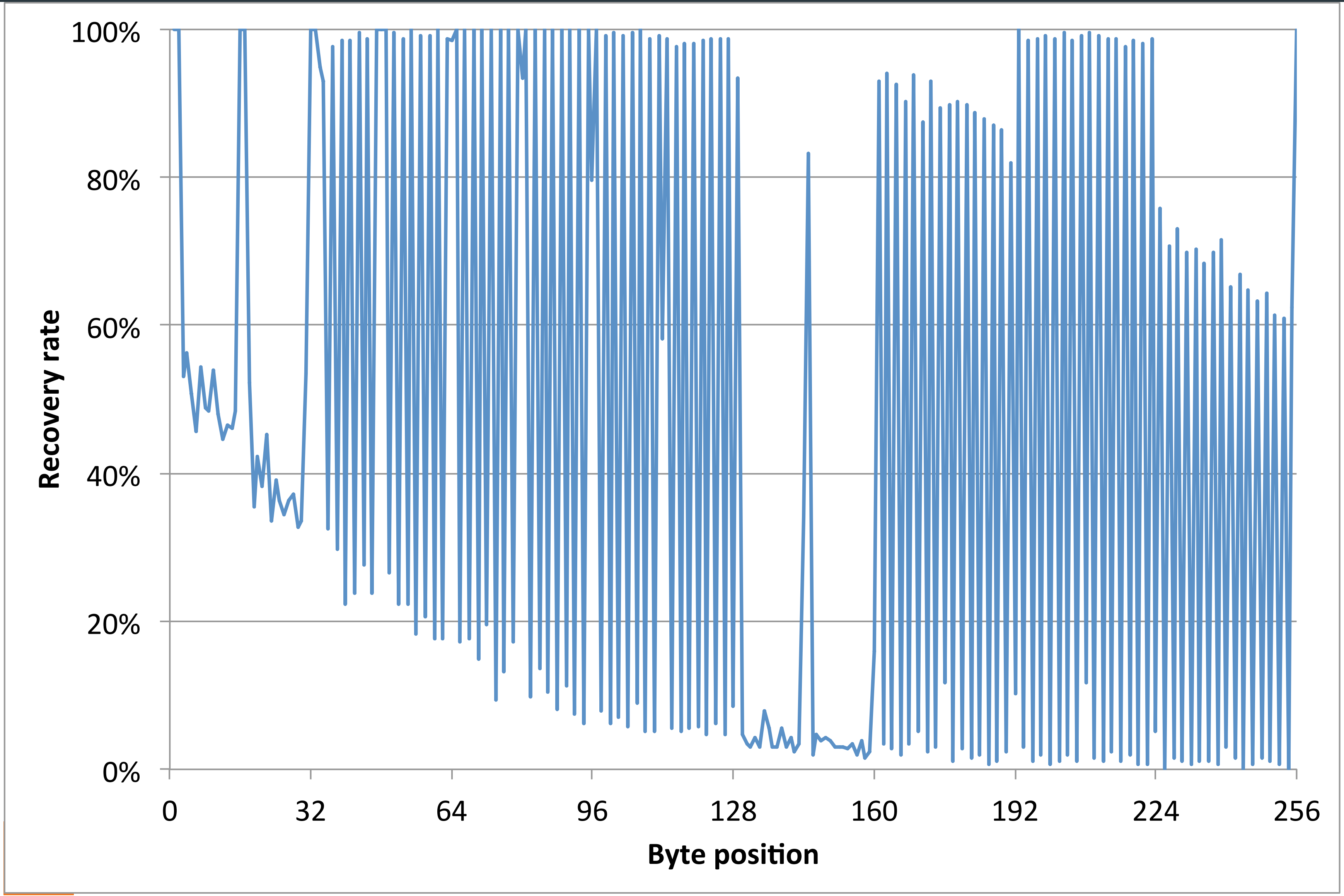
Plaintext recovery based on $\texttt{TSC}_0$ aggregation: 2^{20} frames



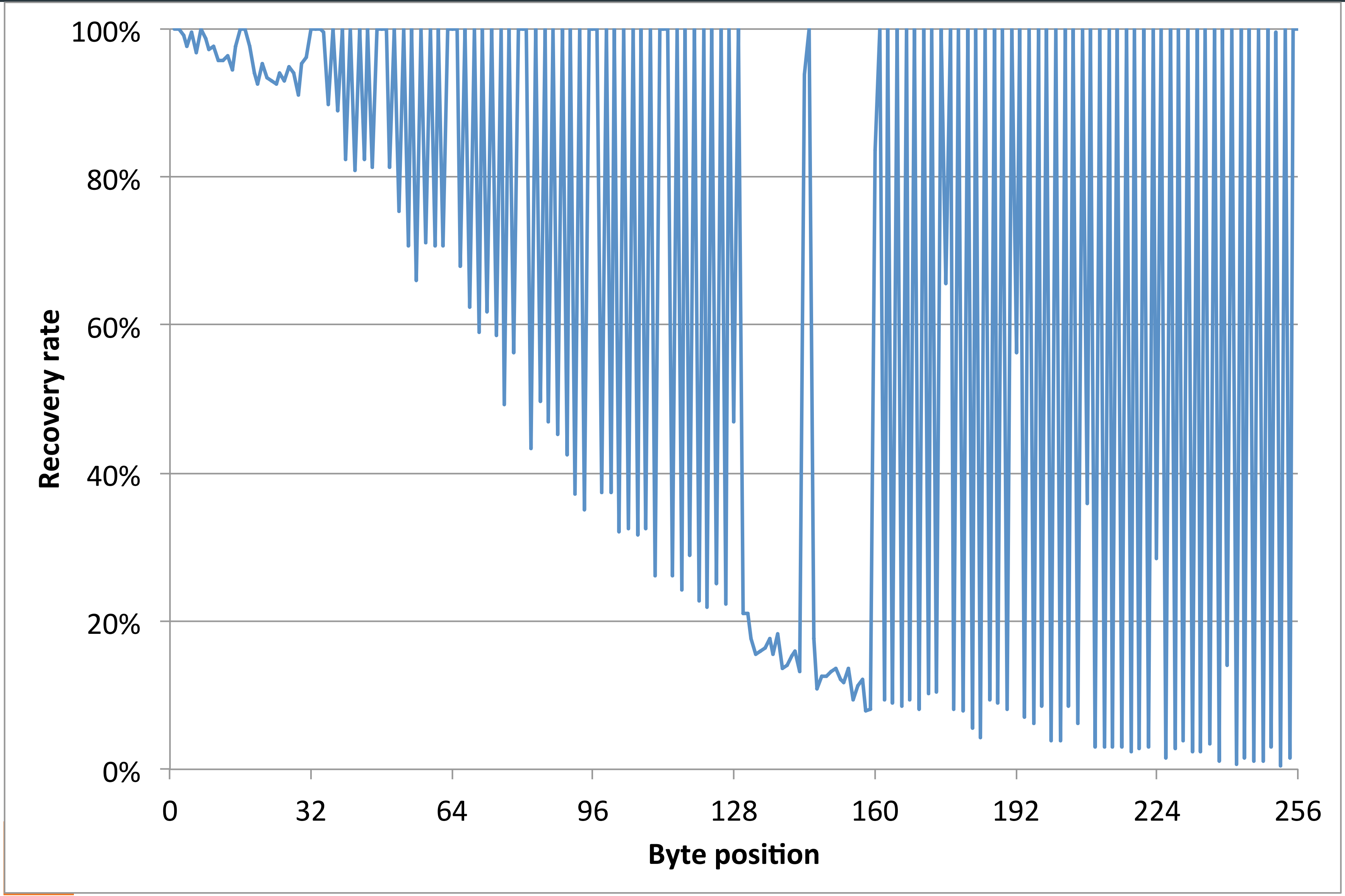
Plaintext recovery based on `TSC`$_0$ aggregation: 2^{22} frames



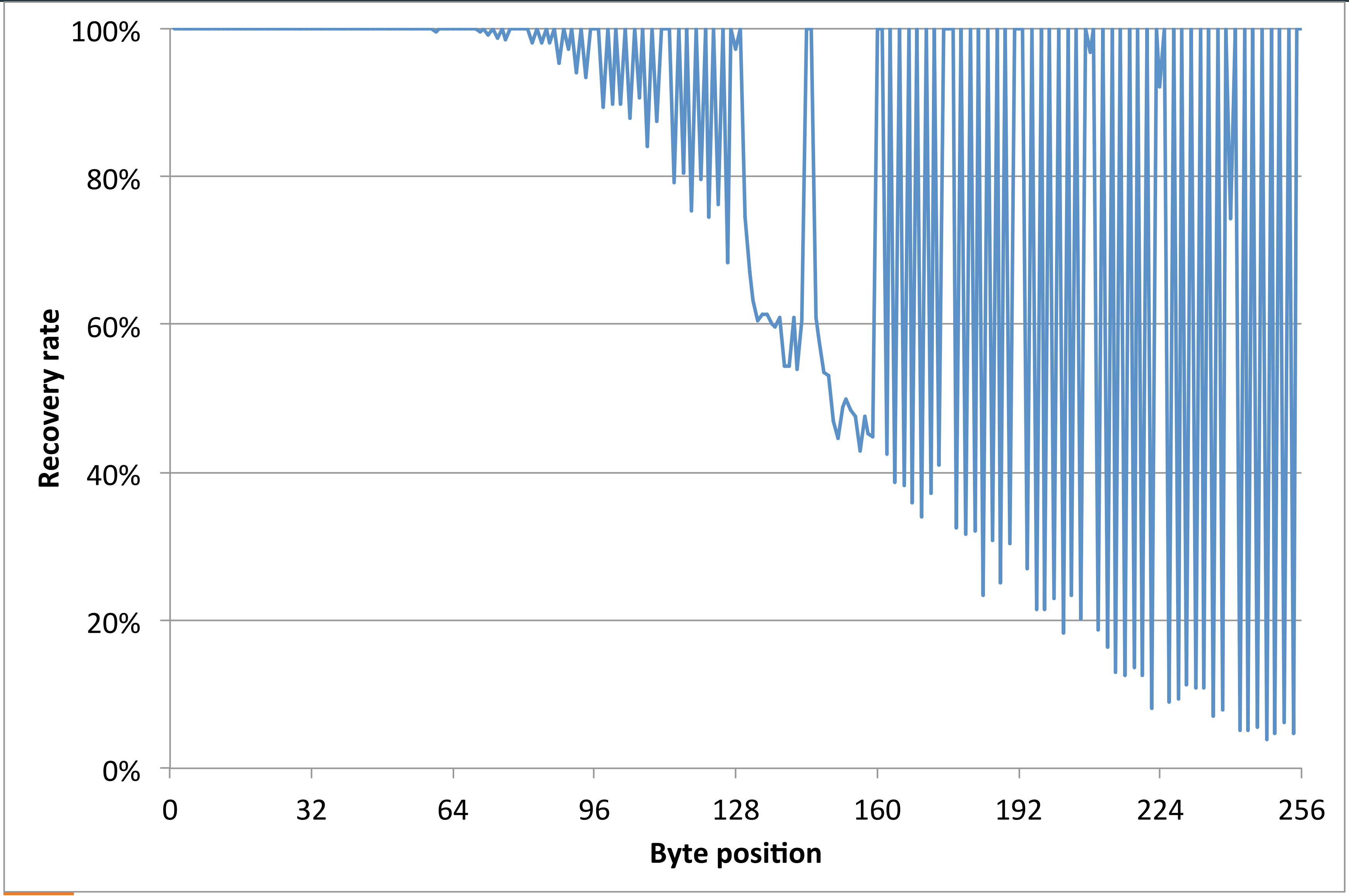
Plaintext recovery based on $\mathtt{TSC}_0$ aggregation: 2^{24} frames



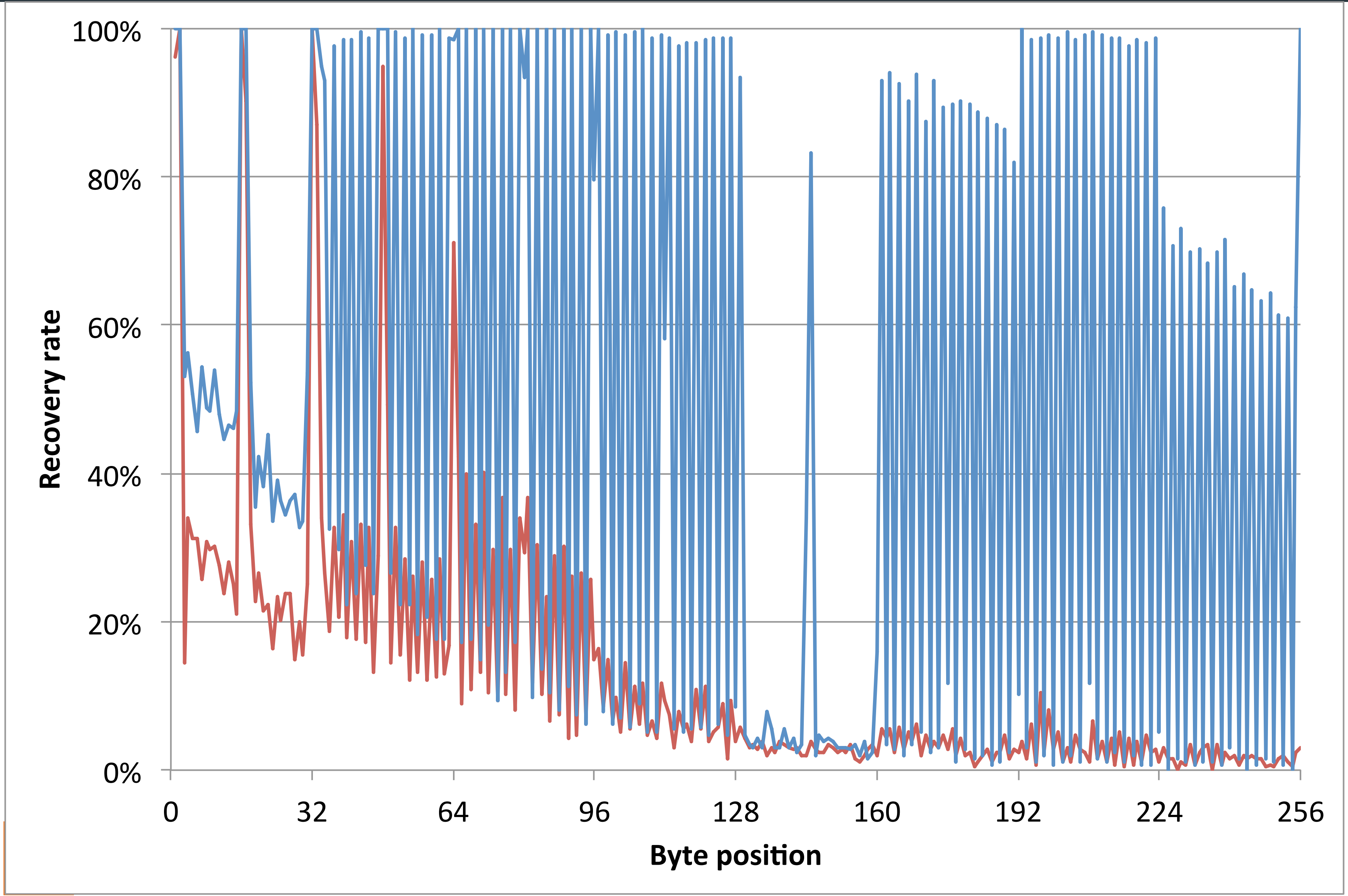
Plaintext recovery based on TSC$_0$ aggregation: 2^{26} frames



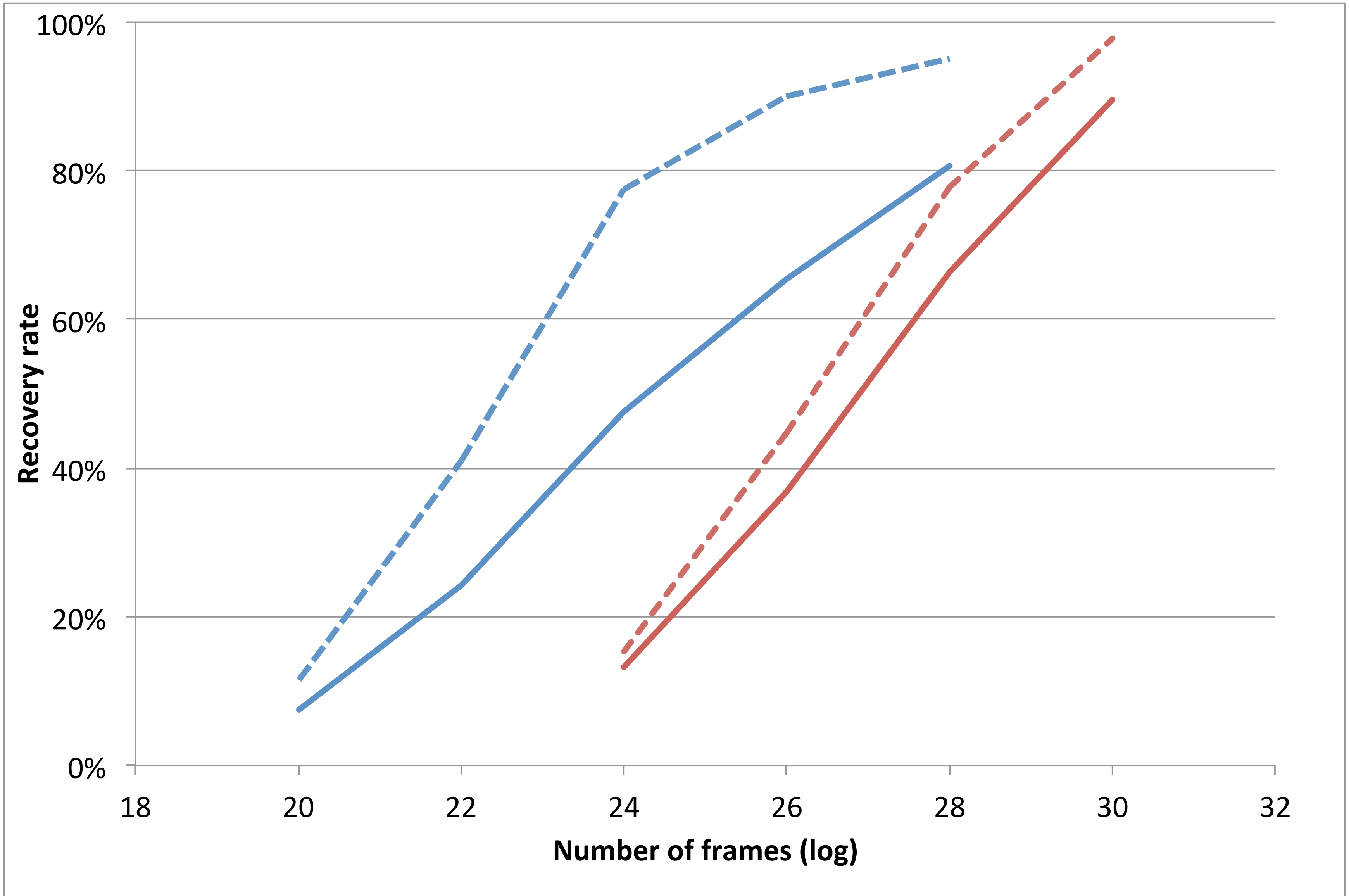
Plaintext recovery based on $\texttt{TSC}_0$ aggregation: 2^{28} frames



Plaintext recovery with $\mathtt{TSC}_0$ aggregation (blue) compared to full aggregation (red): 2^{24} frames



Average recovery rate with $\texttt{TSC}_0$ aggregation (blue) and with full aggregation (red); dash – even positions; solid – all 256 positions

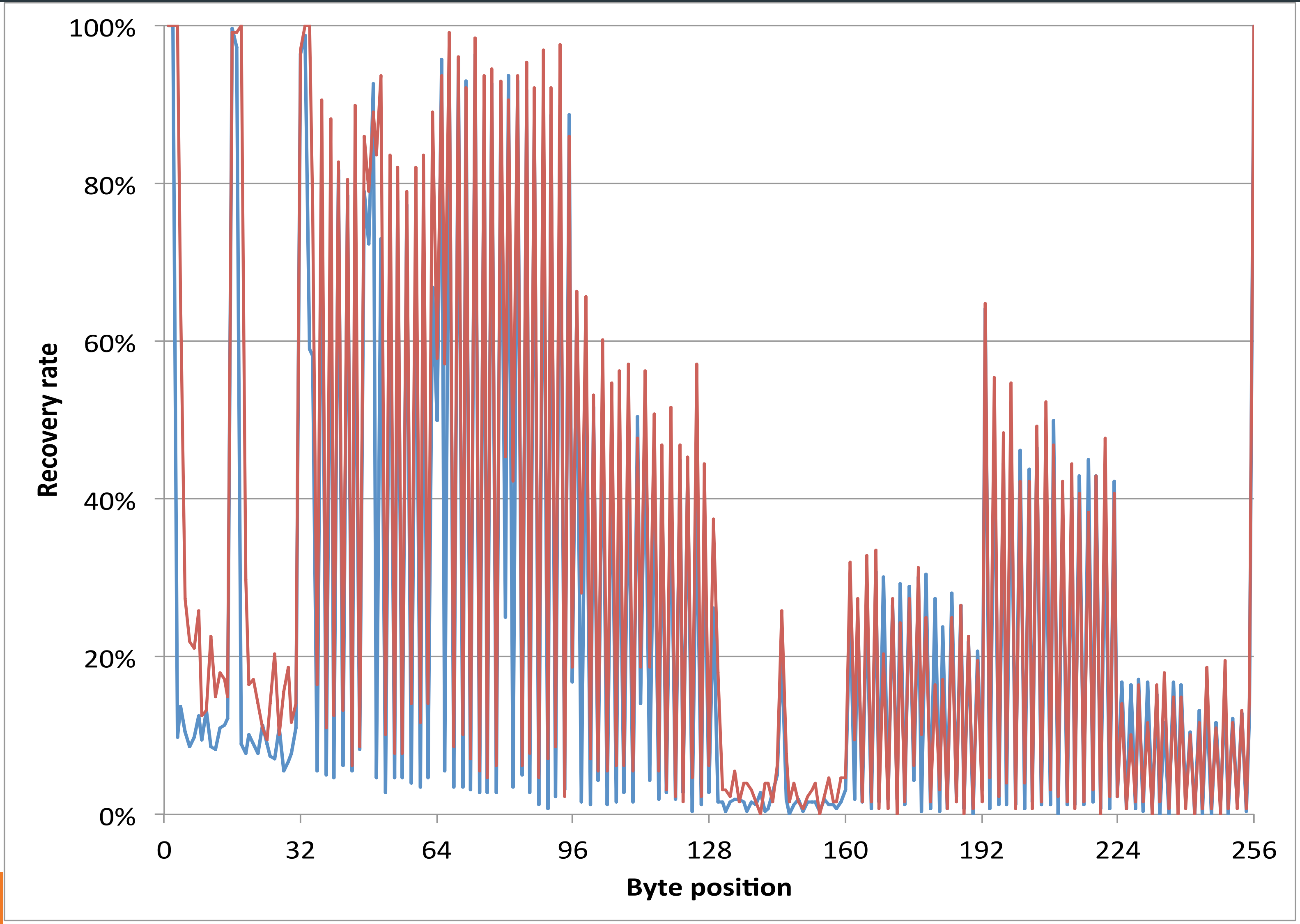


Breaking news!

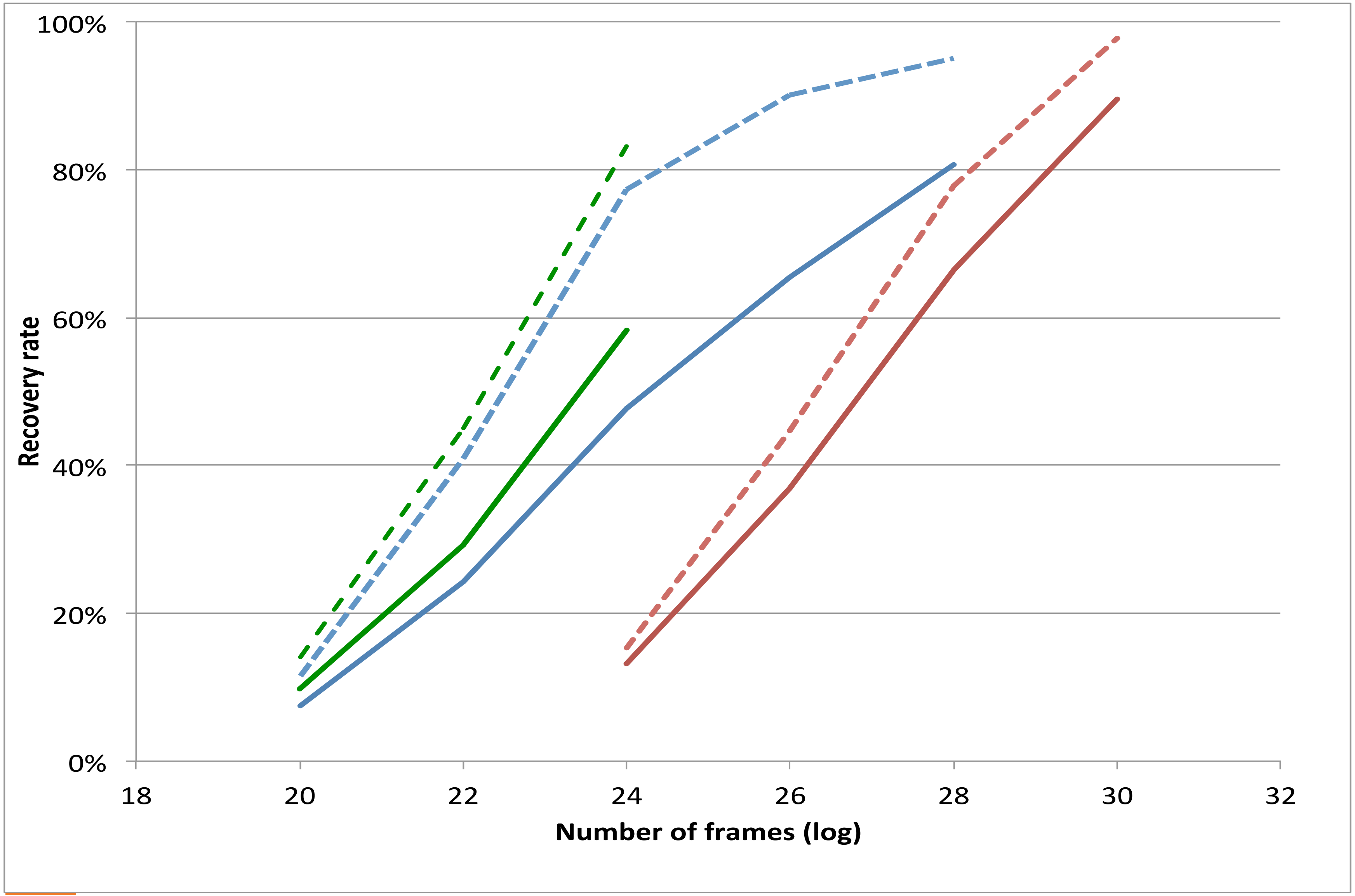
- We have recently carried out a larger-scale keystream bias computation over all ($\mathtt{TSC}_0$,$\mathtt{TSC}_1$) values
  - 2^{32} keystreams per ($\mathtt{TSC}_0$,$\mathtt{TSC}_1$) value, 2^{48} in total.
  - Using Amazon cloud, 256 c3 instances (8192 cores) running for 33 hours, cost of approximately $20k.

- *Slightly* improved plaintext recovery rates compared to previous attacks with $\mathtt{TSC}_0$ aggregation and full aggregation…

Plaintext recovery with `TSC`$_0$ aggregation (blue) compared to no aggregation (red): 2^{22} frames



Average recovery rate with full aggregation (red), `TSC`$_0$ aggregation (blue) and no aggregation (green); dash – even positions; solid – all 256 positions



Concluding remarks/open problems

Concluding remarks

- Plaintext recovery for WPA/TKIP is possible for the first 256 bytes of frames, provided sufficiently many independent encryptions of the same plaintext are available.
- Security is far below the level implied by the 128-bit key `TK`.
- Suitable targets for attack might include fixed but unknown fields in encapsulated protocol headers.
- Targeting HTTP traffic via client-side Javascript also possible, as in TLS attacks.
- Our attack complements known attacks on WPA/TKIP:
  - Passive rather than active (cf. Tews-Beck attack).
  - Ciphertext-only rather than known-plaintext (cf. Sepehrdad-Vaudenay-Vuagnoux attack).
  - Moderate amounts of ciphertext and computation.
  - But repeated plaintext requirement.

Future work and open problems

- Explain all the observed bias behaviour.
  - Some progress has already been made by Sen Gupta-Maitra-Meier-Paul-Sarkar (eprint 2013/476).
  - Not relevant for our plaintext recovery attack, but important for deeper understanding of RC4 in WPA/TKIP and for developing new attacks.

- Try to exploit TSC-specific double-byte biases?
  - We now have an 8 Tbyte dataset (based on 2^{46} keystreams).
  - Attack run-time now becomes significant.

- Study other real-world applications of RC4 in which keys are changed frequently and/or have additional structure.