Protecting encrypted cookies from compression side-channel attacks

Douglas Stebila

Queensland University of Technology

Joint work with Janaka Alawatugoda (QUT) and Colin Boyd (NTNU)

Supported by ARC Discovery Project DP130104304.

Financial Crypto 2015 ▪ IACR eprint 2014/724

Royal Holloway, University of London ▪ January 22, 2015

Introduction

Encryption and compression

Symmetric key encryption

A *symmetric key encryption* scheme is a triple of algorithms:

- KeyGen() –> k
- $\mathrm{Enc}_k(m)$ –> c
- $\mathrm{Dec}_k(c)$ –> m

KeyGen and Enc can be probabilistic

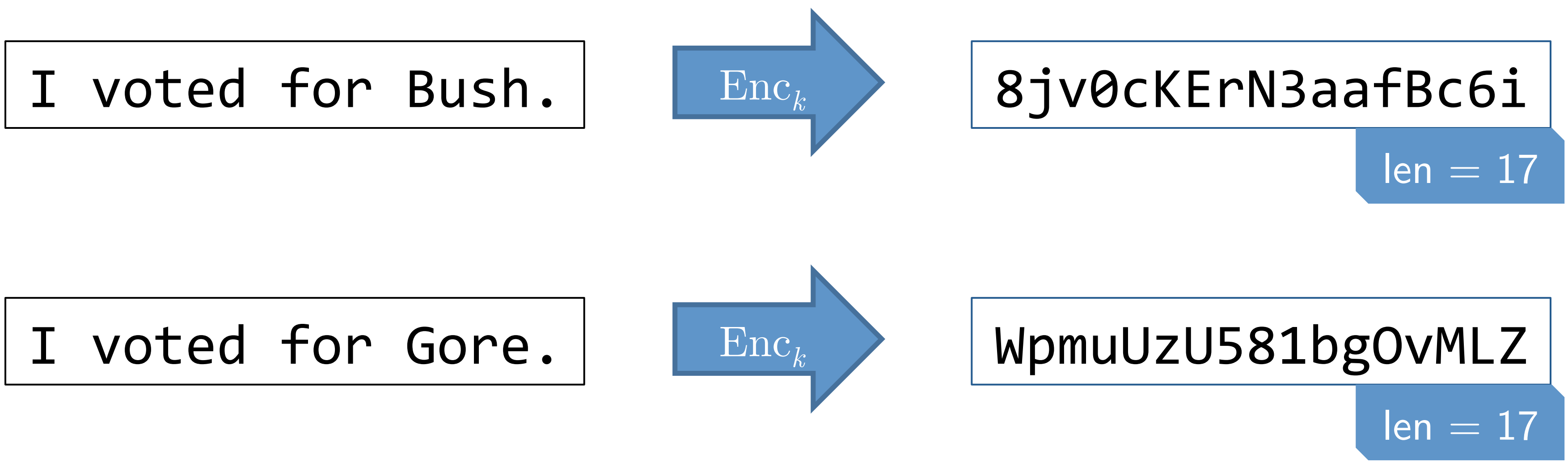
Main security goal:

- **indistinguishability**

Attacker cannot tell apart encryptions of two messages of the same length:

$\mathrm{Enc}_k(m_0)$ looks like $\mathrm{Enc}_k(m_1)$ when $|m_0|=|m_1|$

Symmetric key encryption



same length input => same length output

Compression

A *compression scheme* is a pair of algorithms:

- Comp(m) –> o
- Decomp(o) –> m

Comp may be probabilistic (but usually isn't)

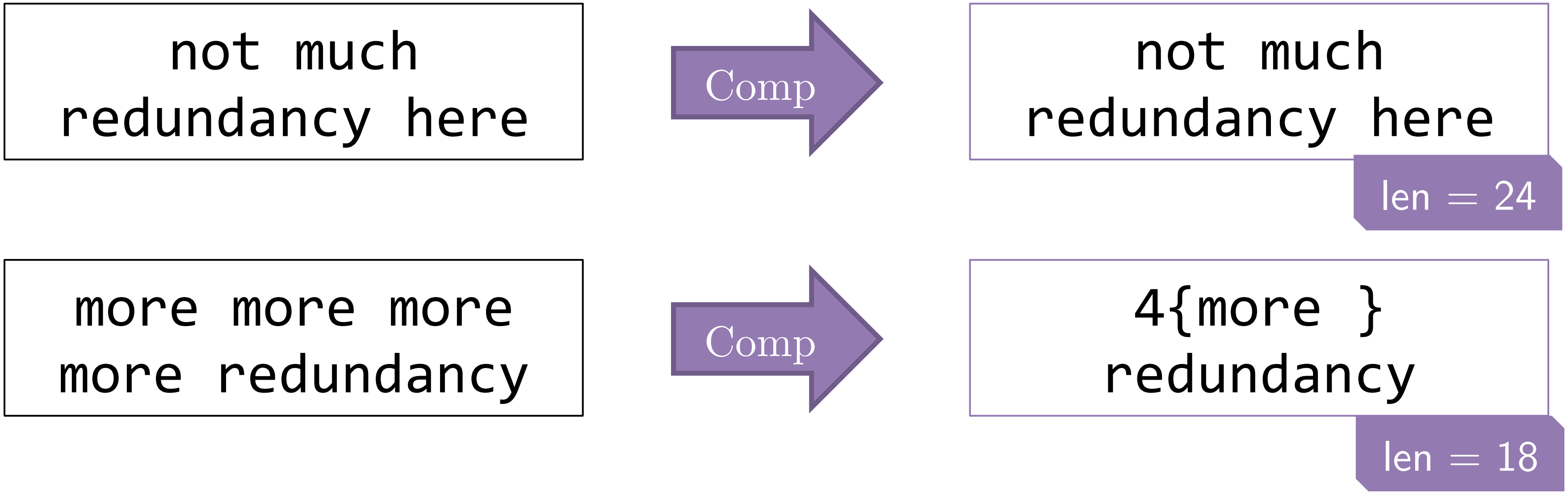
Main security goal:

- **none**

Main functionality goal:

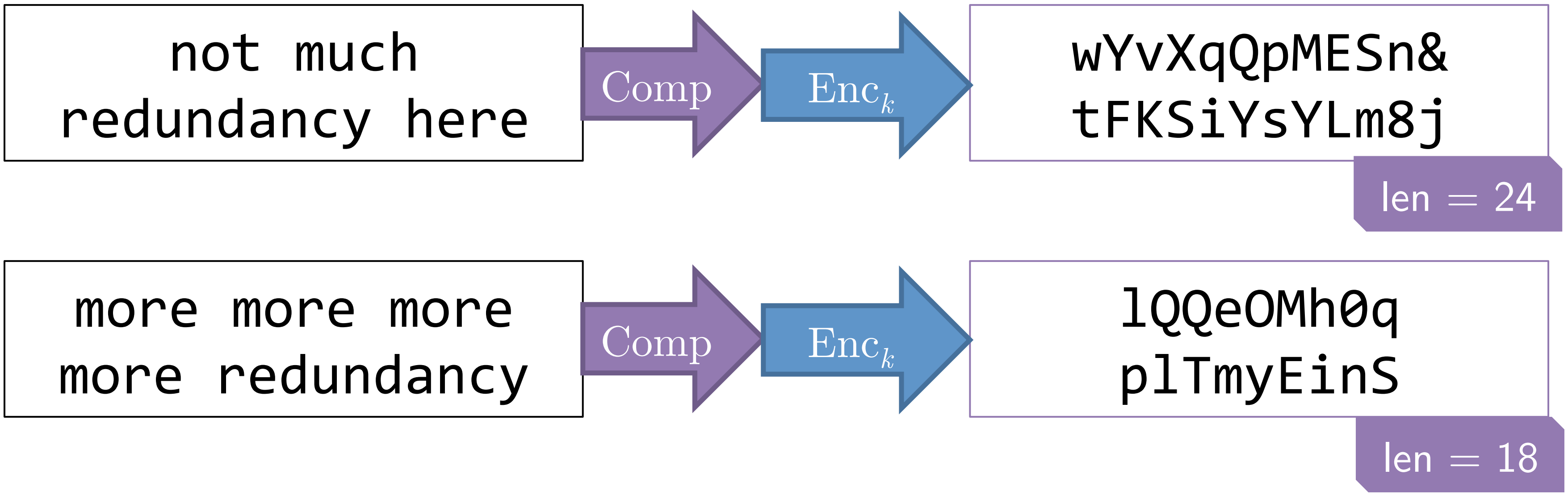
- $|\text{Comp}(m)| << |m|$ for common distribution of m
- Can't be true for all m due to Shannon's theorem

Compression



same length input => possibly different length output

Compression then encryption

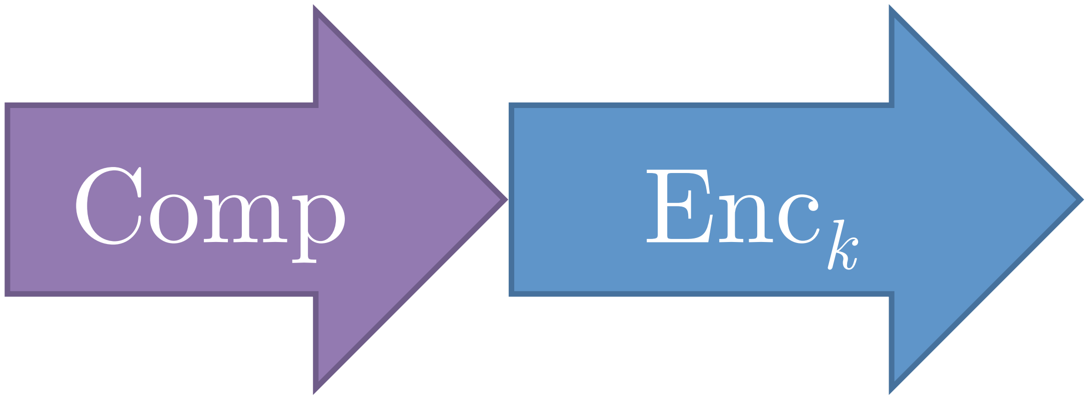


same length input => possibly different length output

A test

Man. U.
2005-2014
loser loser
CHAMP CHAMP
CHAMP loser
CHAMP loser
CHAMP loser

Arsenal
2005-2014
loser loser
loser loser
loser loser
loser loser
loser loser


Comp
Enc_k

Which ciphertext is for which message?

```
yI5pDrFhPk3
15Cmymr6xCb
LTVEAx
```

```
D1fAGUR1zqv
lhXdX3c8qd+
BYBwK6dAnoG
GQGCmvFIM9/
s6WJjgr2
```

One message compresses more

```
Arsenal
2005-2014
10{loser }
```

```
Man. U.
2005-2014
2{loser }
2{CHAMP }
3{CHAMP loser }
```

Deflate (LZ77) compression algorithm

- Replaces repeated strings with back references (distance, length) to previous occurrence.

```
You say potato,
I say potahto.
```

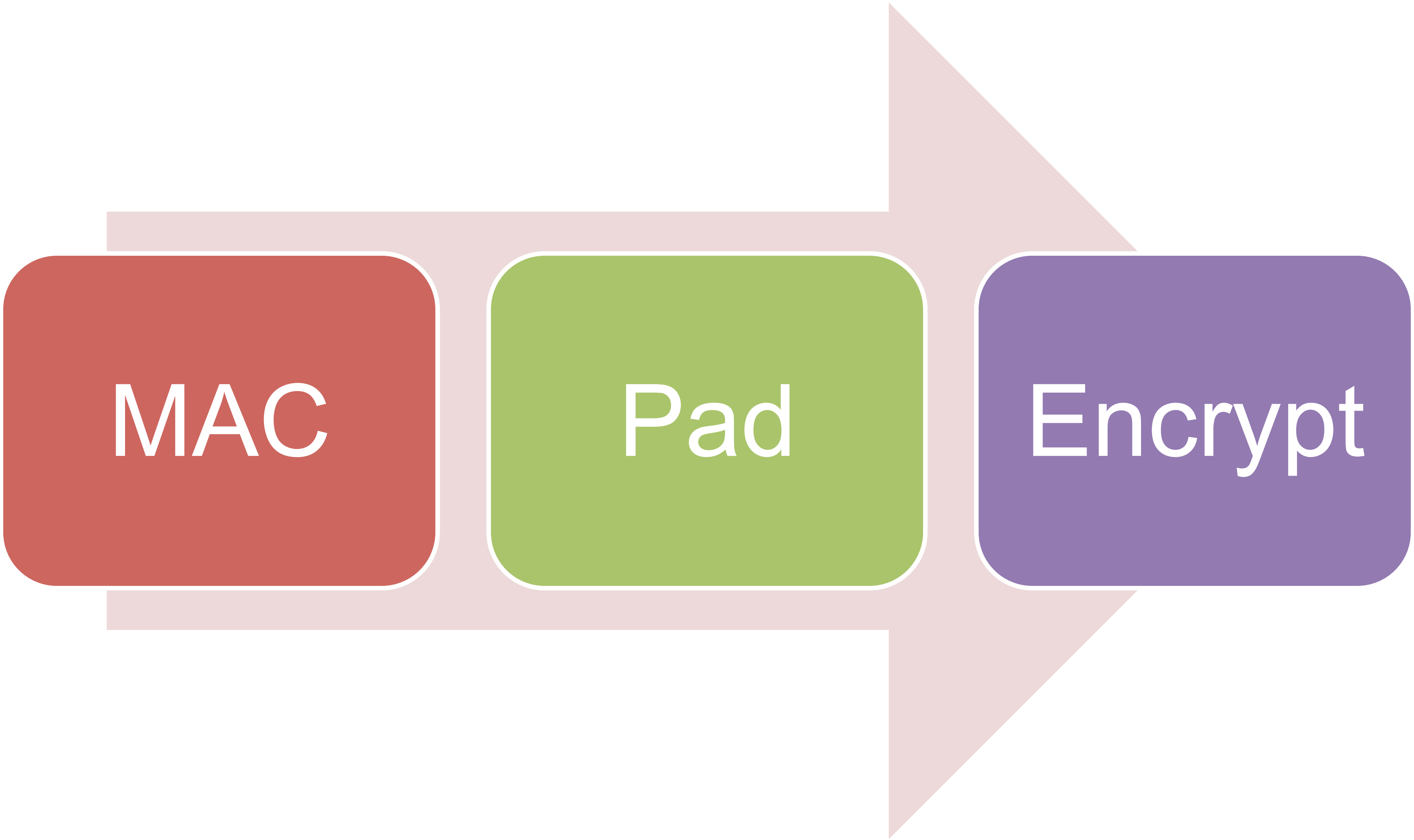
Comp →

```
You say potato,
I (-14,8)hto.
```

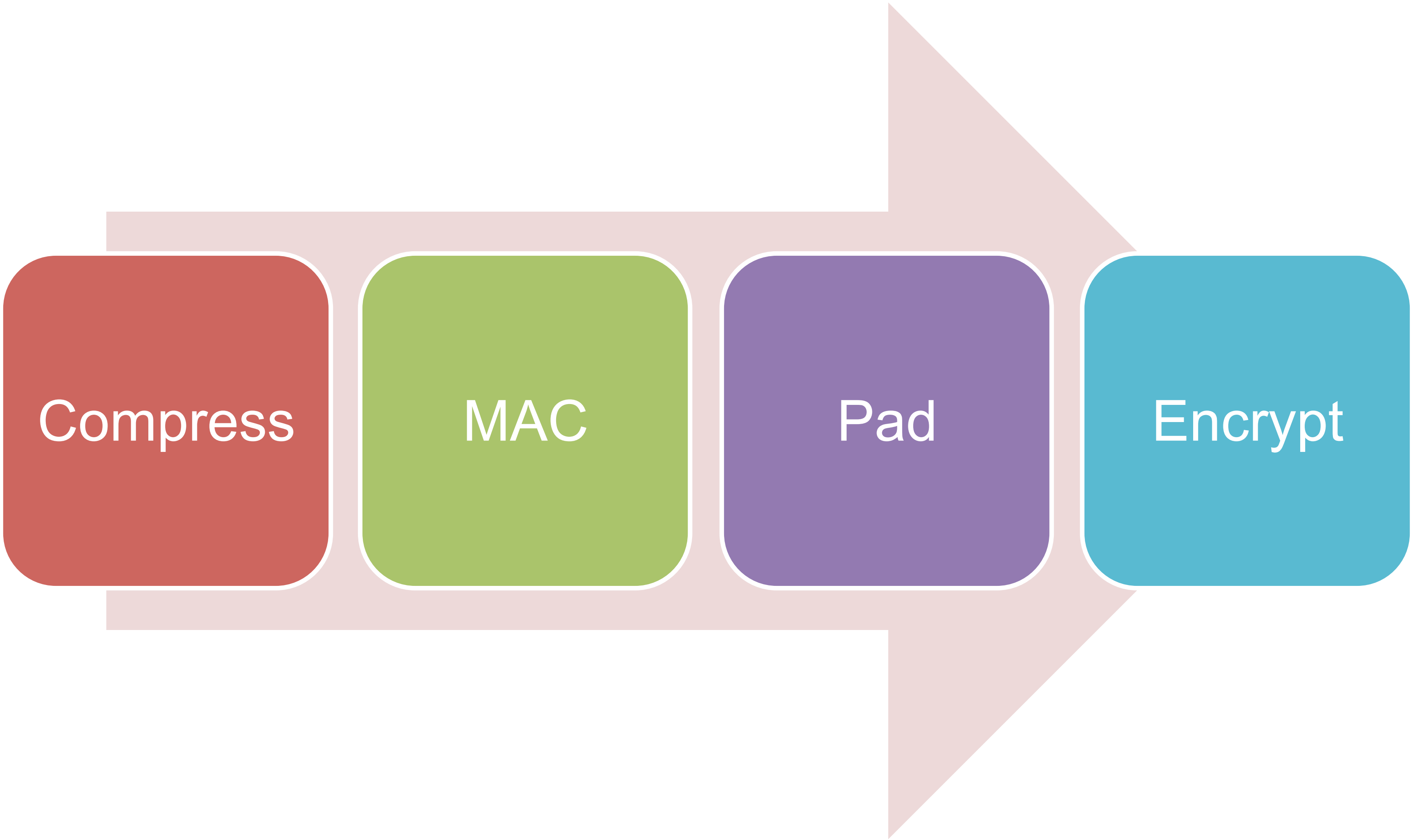
- Important parameter: **window size**
  - How far back does it go to search for occurrences?
  - a.k.a. dictionary size

CRIME attack on compression in TLS

TLS record layer



Compression in TLS record layer

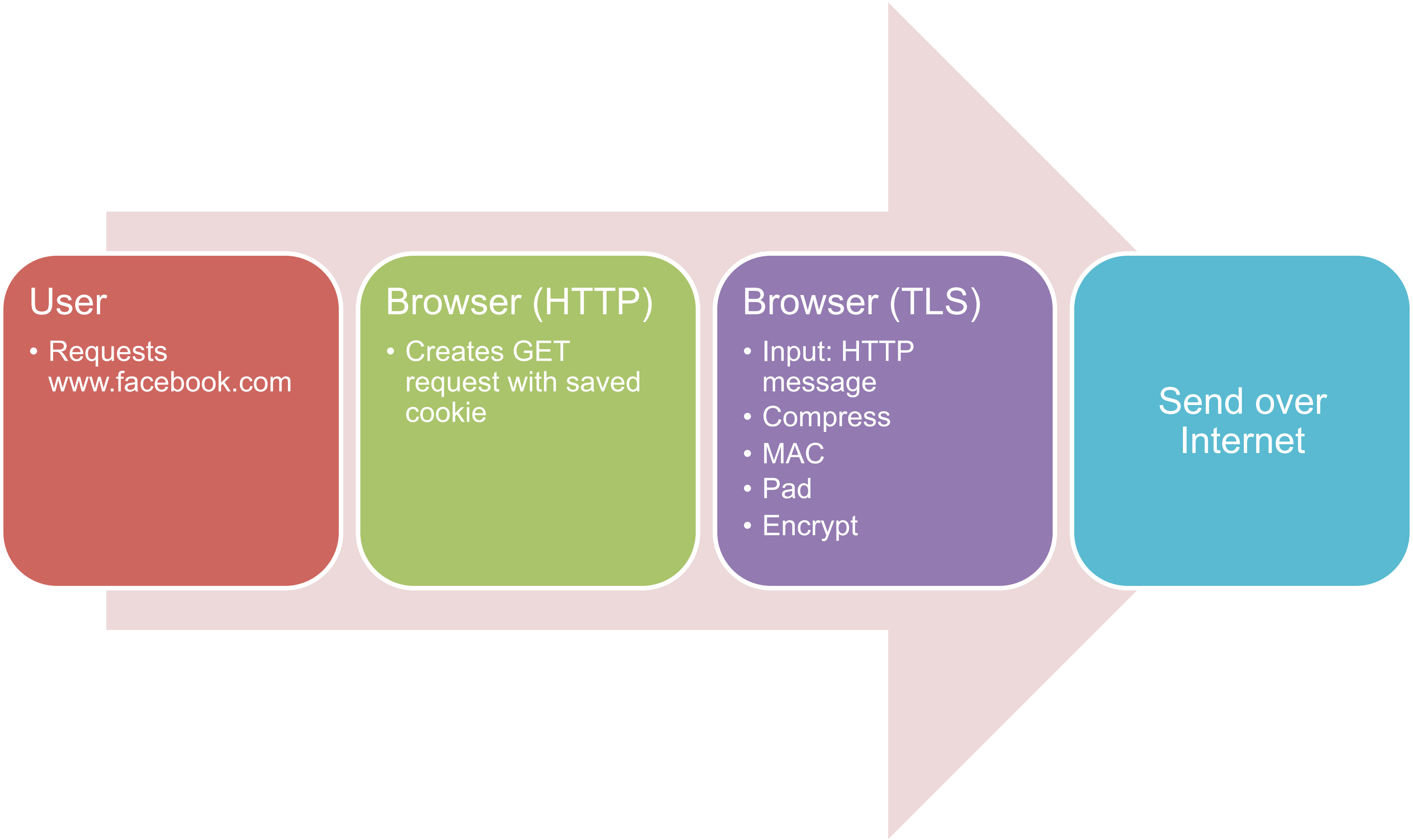


Secret values in HTTP documents

```
GET /
Host: www.facebook.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.10; rv:34.0) Gecko/20100101 Firefox/34.0
Accept: tex                             +xml,application/
xml;q=0.9,*
Accept-Lang
Accept-Enco
DNT: 1
Cookie: datr=DzK9VBnObWDqfL7XLwGSSEsu; reg_fb_ref=https
%3A%2F%2Fwww.facebook.com%2F; reg_fb_gate=https%3A%2F
%2Fwww.facebook.com%2F; dpr=2
Connection: keep-alive
Cache-Control: max-age=0
```

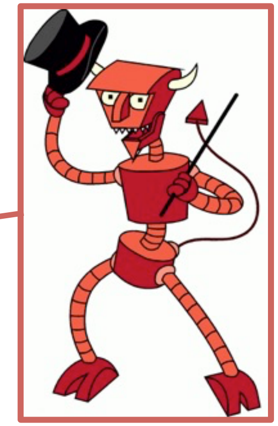
This secret cookie identifies my session to Facebook

Transmitting an HTTP request



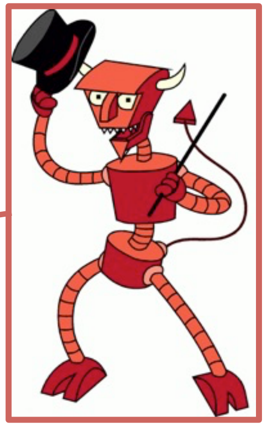
Attack

Please send a GET request for
`https://www.facebook.com/?datr=A`



Attack

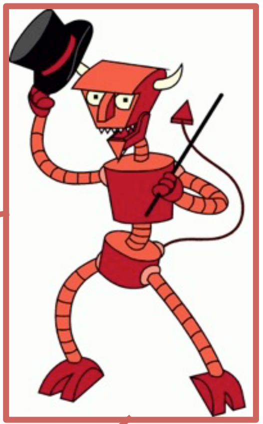
Please send a GET request for
`https://www.facebook.com/?datr=A`




```
GET /?datr=A
Host: www.facebook.com
Cookie: datr=DzK9VBnObW
DqfL7XLwGSSEsu
...
```

Attack

Please send a GET request for
`https://www.facebook.com/?datr=A`



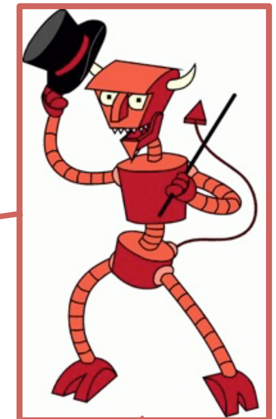
Observes compressed
& encrypted request

VGytgpDn/1Ym5oCdB3Vh2
D5EmdjLRdkx7tEvKG43WJ
yD++cx8CJlBbetQejiXLX
+oQO9bnUMYQwtglOSf9bf
oyWJkYxHsKfqYNqWAfCIg
8U5BK92Ayvk858MJOnTuK

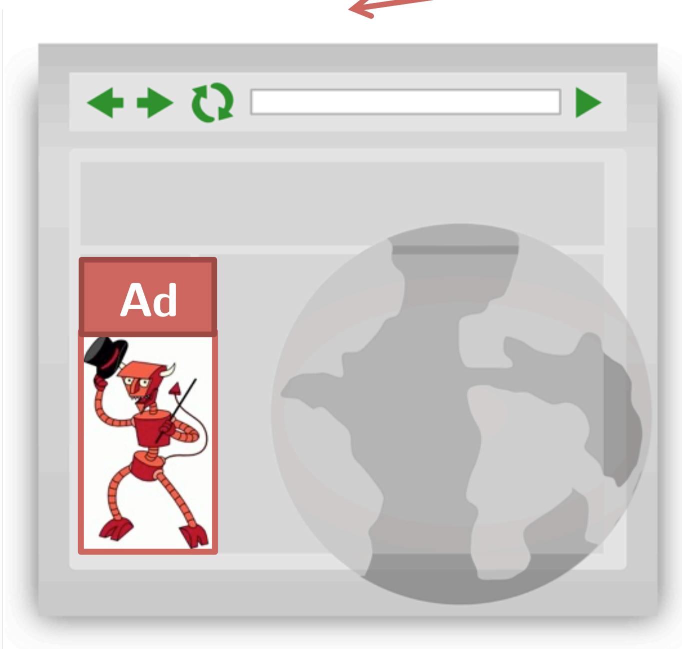
$\text{len} = 204$

Attack

Please send a GET request for
`https://www.facebook.com/?date=B`



Observes compressed
& encrypted request



UQ5ItQ1Y4BVCy37Fhu5K4
hyre7l5P4pWwAYfvnzg9m
R5Qq250PF1yQpf83AFJ34
QS+9BPjUnBzVGENe15r29
rY9tRfIFAdE8ecEmVTFtl
zHy+8EIwxDK67rxM29clJ

len $= 204$

Attack

Please send a GET request for
https://www.facebook.com/?date=C

Observes compressed
& encrypted request

Wdb42n0LeQbVweAoiCZxE
j9O0U+qaGPPbe9Sebz2Dx
GhYWj9U4X0cKYyBpTSpB4
4dOqd4DpCscHEsBdg0p6q
DXiSBJ+MLOKbpRvAAmPhy
9Sn9VPnsHgKyB4I1lgCKA

len = 204

Attack

Please send a GET request for
https://www.facebook.com/?date=D

Observes compressed
& encrypted request

O8Gb8JwSuoNrcQ7190KSs
nM7n22lOtByzmvv555ZP+
+4lNW2wIuRrTF6KlKdjOB
425VVDUbKKdHNF9YaaxTy
lVWBVo1ApZ4PTSnB1J0pt
jAsecGXjRXOXTwye

len = 199

Attack

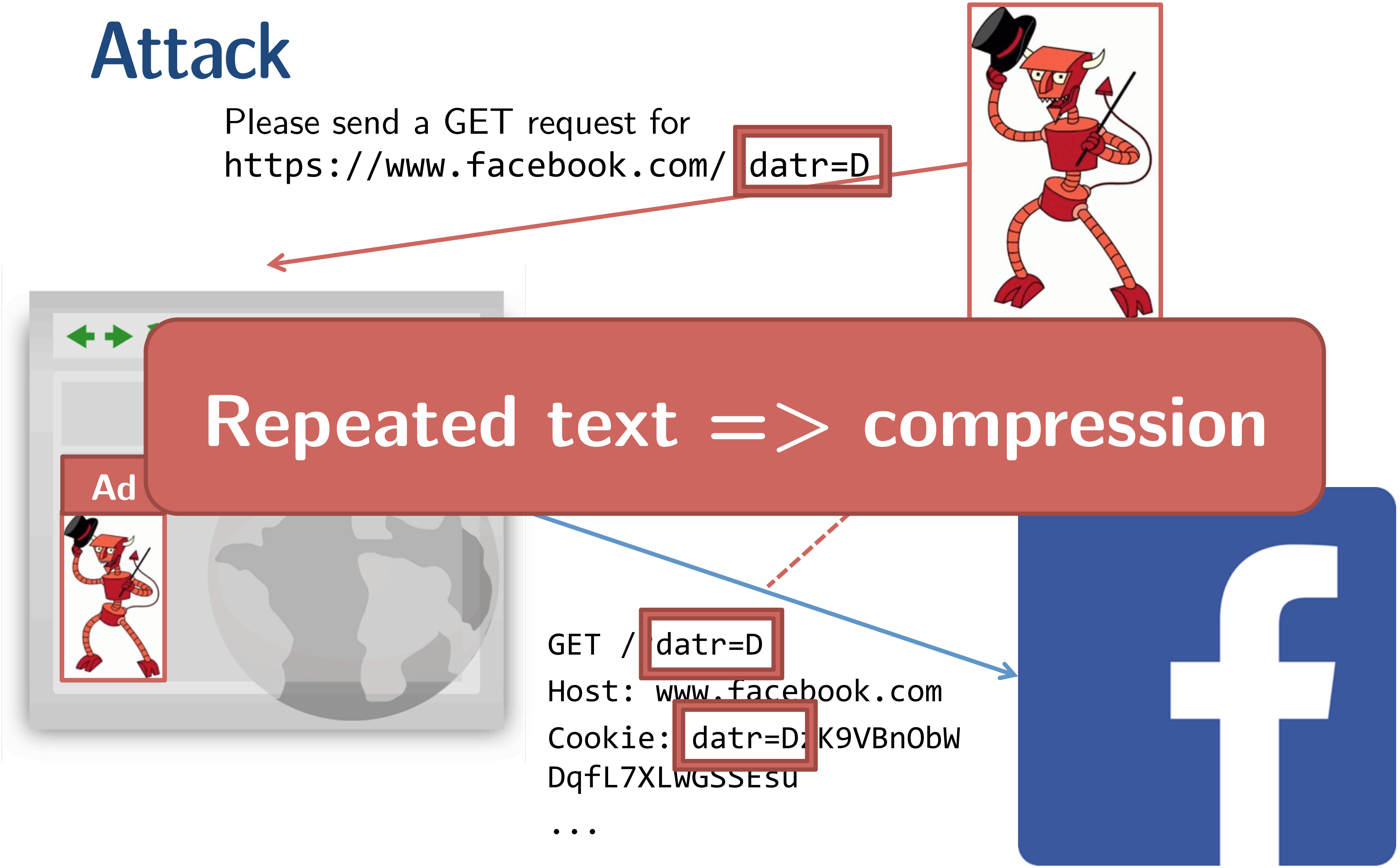
Please send a GET request for
`https://www.facebook.com/?datr=Da`

Observes compressed
& encrypted request


```
Ok3MV18blnYFIjz2tcucQ
x2mJ8MLULVqMSYO9Lo1r0
wxwjEG8pLwaPaVtrnf46l
ypdqbYQ22oJw63ixkS1HR
QVfz8UKs9tOhPvTAwUiwS
yukxrKq9x9I+3fO8lv8aU
```


len $=$ 205

Attack


Please send a GET request for
https://www.facebook.com/ datr=D
Ad
Repeated text => compression
GET / datr=D
Host: www.facebook.com
Cookie: datr=D K9VBnObW
DqfL7XLWGSSEsu
...

CRIME attack on TLS "Compression Ratio Info-leak Made Easy"

- "Rizzo and Duong [ekoparty 2012]

- Victim visits adversary-controlled page
- Adversarial Javascript causes browser to make many requests
- Figure out 1st letter of cookie
- Figure out 2nd letter of cookie
- Figure out 3rd letter of cookie
- ...

A few tricky bits to make it work in TLS:

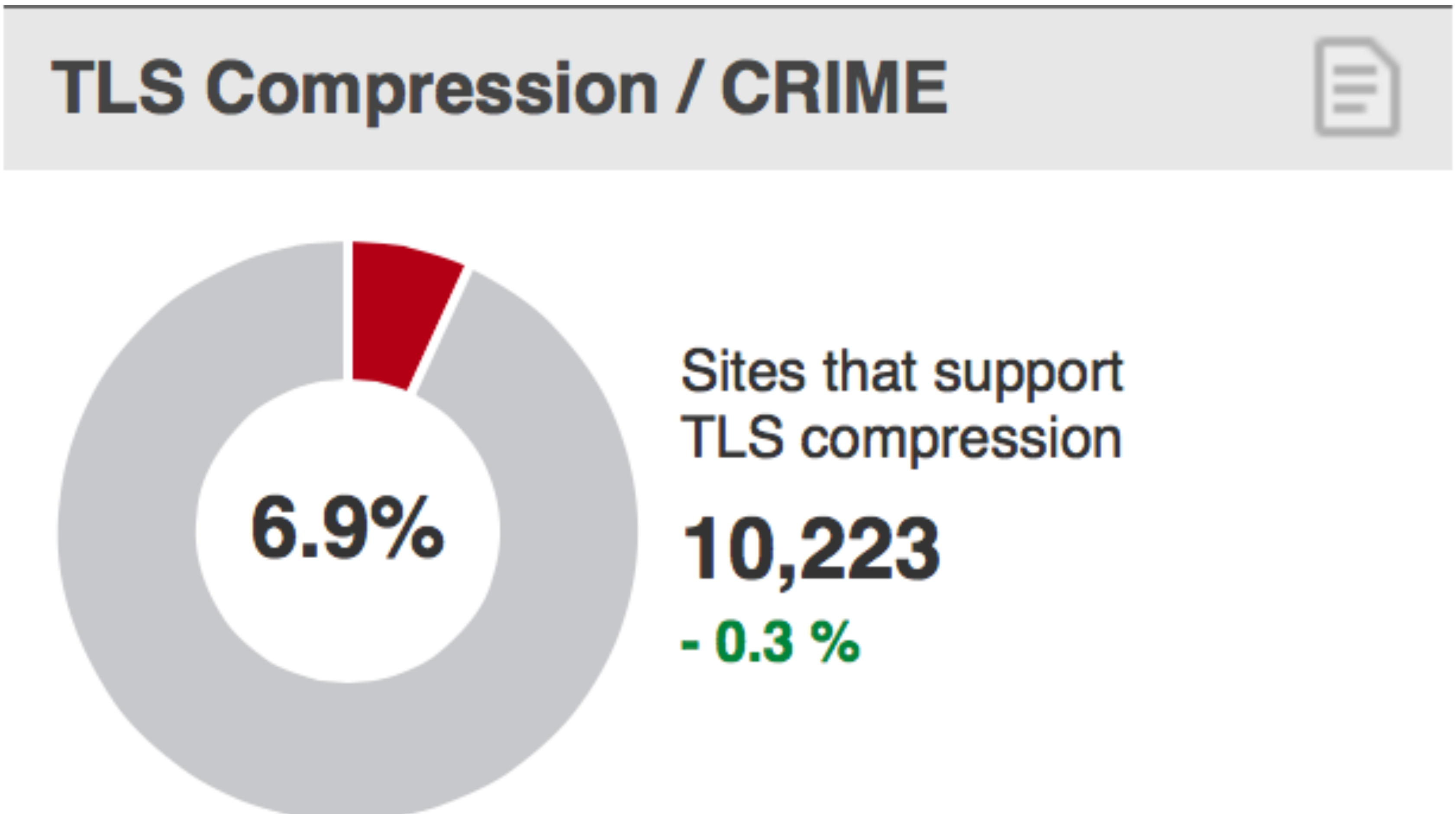
- TLS splits plaintext into 16K records then compresses and encrypts each record separately
- Need to ensure that you can observe length differences based on compression

- But it can be made to work!

CRIME wasn't new

- Kelsey [FSE 2002] theorized length-based attacks on compression-encryption with adversary-chosen prefix.

Impact of CRIME attack



January 7, 2015 • https://www.trustworthyinternet.org/ssl-pulse/

But...

- Compression is present elsewhere on the Internet.
- HTTP allows gzip compression of the body

BREACH attack on compression in HTTP

BREACH attack

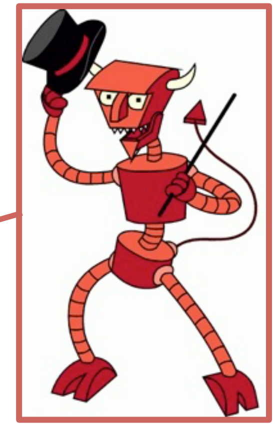
- Attack against HTTP compression hypothesized in CRIME presentation

"Browser Reconnaissance and Exfiltration via Adaptive Compression of Hypertext"

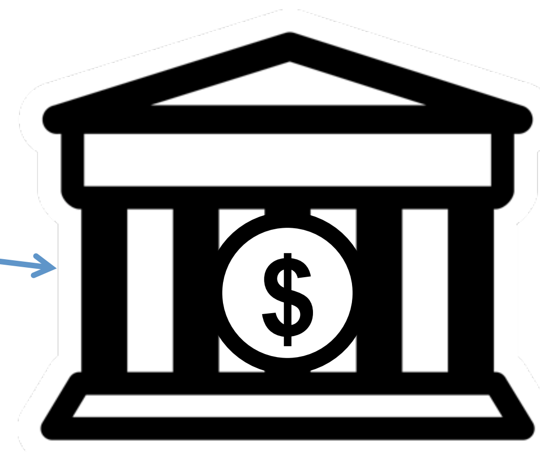
- attack demonstrated against secrets in HTML
- Gluck, Harris, Prado [Black Hat 2013]

Cross-site request forgery

Please send a GET request for
`https://www.bank.com/transfer`
`?to=Eve&amount=1000000`



```
GET /transfer?to=Eve
&amount=1000000
Host: www.bank.com
Cookie: account=Alice
...
```



Anti-CSRF tokens

Protection strategy: server hides a random token in each HTML form it creates and will only execute action if received form contains that token

```
<form action="/money_transfer" method="post">
<input type="hidden" name="csrftoken"
       value="OWT4NmQlODE4ODRjN2Q1NTlhMmZlYWE...">
...
</form>
```

BREACH Attack

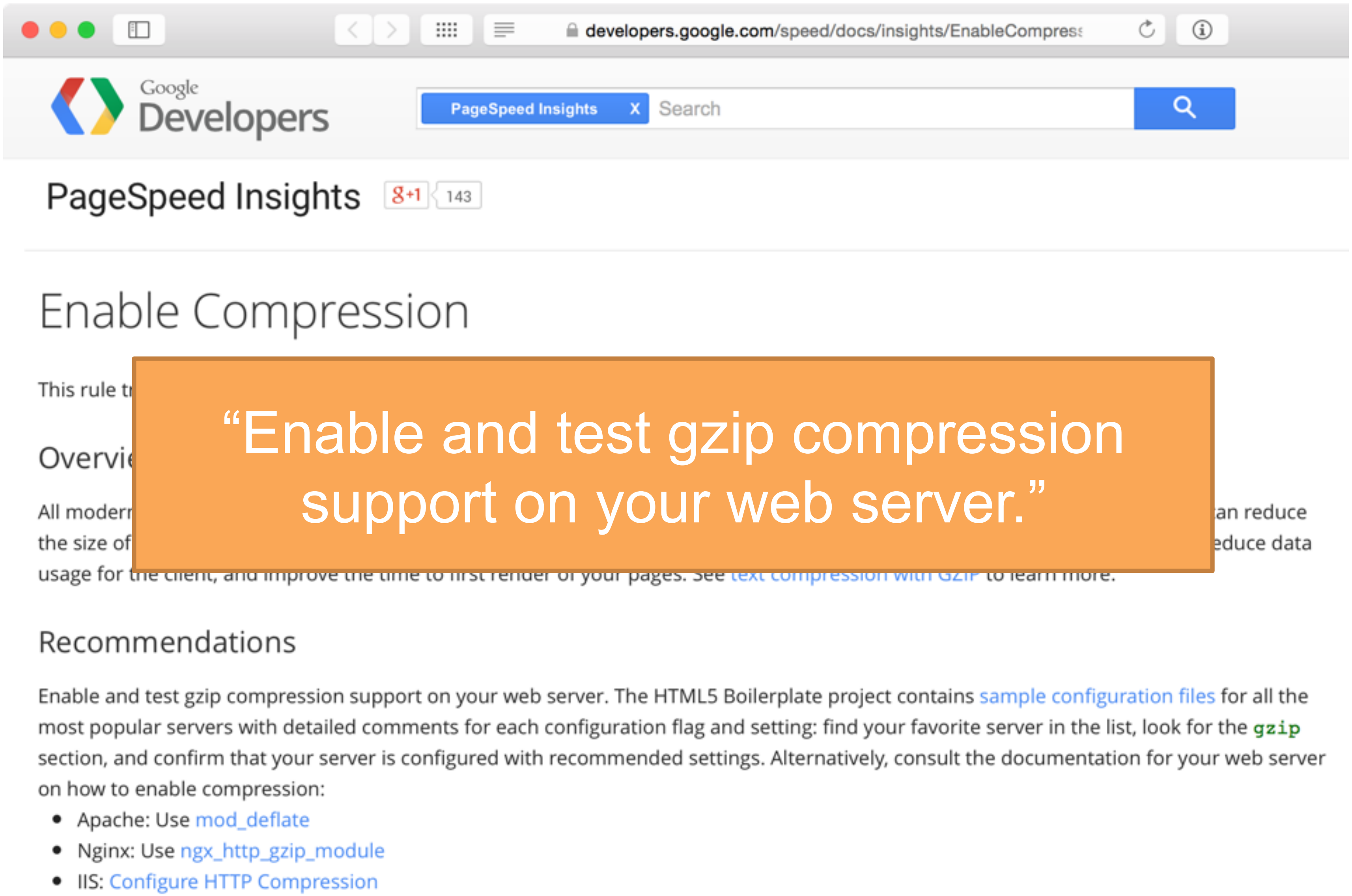
Works against websites that echo user input in the same page as a valuable secret (e.g., anti-CSRF token)

```
<p>Welcome, <?=$_GET['username']?>.</p>
<form action="/money_transfer" method="post">
<input type="hidden" name="csrftoken"
       value="OWT4NmQ1ODE4ODRjN2Q1NTlhMmZlYWE...">
...
</form>
```

Recommendations from BREACH attack

1. Disabling HTTP compression
2. Separating secrets from user input
3. Randomizing secrets per request
4. Masking secrets (effectively randomizing by XORing with a random nonce)
5. Length hiding (by adding a random number of bytes to the responses)
6. Rate-limiting the requests

Impact of BREACH attack



Compression in network protocols

HTTP/1.1	SPDY	HTTP/2	Others
▪ supports compression ▪ BREACH attack ▪ still widely used	▪ supports compression ▪ CRIME/ BREACH work against early versions	▪ separate compression of every headers ▪ uses special algorithm HPACK for header compression	▪ SSH ▪ PPTP ▪ OpenVPN ▪ XMPP ▪ IMAP ▪ SMTP ▪ (see CRIME slides)

Recommendations from BREACH attack

1. Disabling HTTP compression
2. Separating secrets from user input
3. Randomizing secrets per request
4. Masking secrets (effectively randomizing by XORing with a random nonce)
5. Length hiding (by adding a random number of bytes to the responses)
6. Rate-limiting the requests
7. Use non-adaptive compression algorithm

Security definitions

Encryption security: IND-CPA

$\underline{\mathrm{Exp}_{\Pi}^{\textsf{IND-CPA}}(\mathcal{A})}$

$k \xleftarrow{\$} \Pi.\mathrm{KeyGen}()$

$b \xleftarrow{\$} \{0,1\}$

$(m_0, m_1, st) \xleftarrow{\$} \mathcal{A}^E()$

if $|m_0| \neq |m_1|$, **then return** $\perp$

$c \leftarrow \Pi.\mathrm{Enc}_k(m_b)$

$b' \xleftarrow{\$} \mathcal{A}^E(c, st)$

return $(b' = b)$

$\underline{E(m)}$

return $\Pi.\mathrm{Enc}_k(m)$

Entropy-restricted encryption security: ER-IND-CPA [KelTam14]

$\mathrm{Exp}^{\text{ER-IND-CPA}}_{\Pi,\mathcal{L}}(\mathcal{A})$

1: $k \xleftarrow{\$} \Pi.\mathrm{KeyGen}()$
2: $b \xleftarrow{\$} \{0,1\}$
3: $(m_0, m_1, st) \xleftarrow{\$} \mathcal{A}^E()$
4: **if** $m_0 \notin \mathcal{L}$ or $m_1 \notin \mathcal{L}$, **then return** $\perp$
5: $c \leftarrow \Pi.\mathrm{Enc}_k(m_b)$
6: $b' \xleftarrow{\$} \mathcal{A}^E(c, st)$
7: **return** $(b' = b)$

$$\mathcal{L} = \mathcal{L}_\ell = \{m \in \mathcal{M} : |\mathrm{Comp}(m)| = \ell\}$$

Kelsey/CRIME = Adaptive chosen prefix/suffix attack

- There is a secret value ck.
- Attacker can adaptively choose values m', m'' and receive

$$\mathrm{Enc}_k(\mathrm{Comp}(m' \mid\mid ck \mid\mid m''))$$

- Attacker's goal is to learn something about ck

New security definitions

Attacker's powers

Adaptively obtain encryptions of

$$m' \mathbin{||} ck \mathbin{||} m''$$

for m', m'' of the adversary's choice

Attacker's goals

- **Cookie recovery**: fully recover the secret cookie ck
- **Chosen cookie indistinguishability**: distinguish which of two chosen cookies ck_0, ck_1 is used
- **Random cookie indistinguishability**

Cookie-recovery (CR) security

$\underline{\mathrm{Exp}^{\mathsf{CR}}_{\Psi,\mathcal{CK}}(\mathcal{A})}$

1: $k \xleftarrow{\$} \Psi.\mathrm{KeyGen}()$
2: $ck \xleftarrow{\$} \mathcal{CK}$
3: $ck' \xleftarrow{\$} \mathcal{A}^{E_1,E_2}()$
4: **return** $(ck' = ck)$

$\underline{E_1(m', m'')}$

1: **return** $\Psi.\mathrm{Enc}_k(m' \| ck \| m'')$

$\underline{E_2(m)}$

1: **return** $\Psi.\mathrm{Enc}_k(m)$

Goal: fully recover the secret cookie *ck*.

- Models an attacker who is trying to steal a secret value to use
  - e.g. CRIME/BREACH
- Does not provide confidentiality of other parts of plaintext

Chosen cookie indistinguishability (CCI)

$\underline{\mathrm{Exp}^{\mathsf{CCI}}_{\Psi,\mathcal{CK}}(\mathcal{A})}$

1: $k \xleftarrow{\$} \Psi.\mathrm{KeyGen}()$
2: $(ck_0, ck_1, st) \xleftarrow{\$} \mathcal{A}^{E_2}()$ s.t. $|ck_0| = |ck_1|$
3: $b \xleftarrow{\$} \{0,1\}$
4: $b' \xleftarrow{\$} \mathcal{A}^{E_1,E_2}(ck_0, ck_1, st)$
5: **return** $(b' = b)$

$\underline{E_1(m', m'')}$

1: **return** $\Psi.\mathrm{Enc}_k(m' \| ck_b \| m'')$

$\underline{E_2(m)}$

1: **return** $\Psi.\mathrm{Enc}_k(m)$

Goal: determine which of two **chosen** cookies ck_0, ck_1 is used throughout

- Models an attacker who is trying to learn about cookies used
  - e.g., passive surveillance
- Does not provide confidentiality of other parts of plaintext

Random cookie indistinguishability (RCI)

$\underline{\mathrm{Exp}^{\mathsf{RCI}}_{\Psi,\mathcal{CK}}(\mathcal{A})}$

1: $k \xleftarrow{\$} \Psi.\mathrm{KeyGen}()$

2: $(ck_0, ck_1) \xleftarrow{\$} \mathcal{CK}$
s.t. $|ck_0| = |ck_1|$; $st \leftarrow \perp$

3: $b \xleftarrow{\$} \{0,1\}$

4: $b' \xleftarrow{\$} \mathcal{A}^{E_1,E_2}(ck_0, ck_1, st)$

5: **return** $(b' = b)$

$\underline{E_1(m', m'')}$

1: **return** $\Psi.\mathrm{Enc}_k(m' \| ck_b \| m'')$

$\underline{E_2(m)}$

1: **return** $\Psi.\mathrm{Enc}_k(m)$

Goal: determine which of two **random** cookies ck_0, ck_1 is used throughout

- Intermediate notion, possibly still relevant
- Does not provide confidentiality of other parts of plaintext

Relations and separations

$$\text{CCI} \implies \text{RCI} \implies \text{CR}$$

$$\text{CR} \not\Longrightarrow \text{RCI} \not\Longrightarrow \text{CCI}$$

$$\text{ER-IND-CPA} \implies \text{IND-CPA} \implies \text{CCI}$$

$$\text{CCI} \not\Longrightarrow \text{IND-CPA}$$

Compressing encryption

Definitions shown are all about encryption schemes.

- A compressing encryption scheme *is* an encryption scheme.

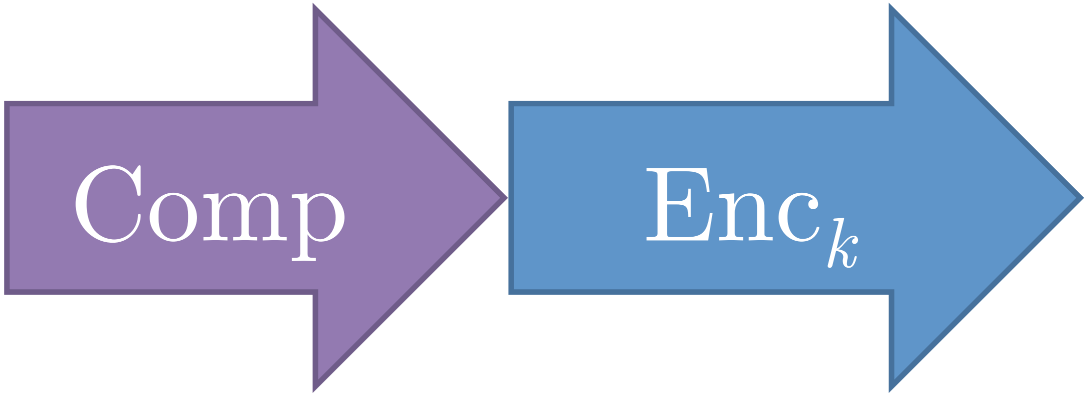
The natural compression-encryption scheme

Let $\Gamma = (\mathrm{Comp}, \mathrm{Decomp})$ be a compression scheme.

Let $\Pi = (\mathrm{KeyGen}, \mathrm{Enc}, \mathrm{Dec})$ be a symmetric-key encryption scheme.

The symmetric-key compression-encryption scheme $\Pi \circ \Gamma$ is:

$$(\Pi \circ \Gamma).\mathrm{KeyGen}() = \Pi.\mathrm{KeyGen}()$$
$$(\Pi \circ \Gamma).\mathrm{Enc}_k(m) = \Pi.\mathrm{Enc}_k(\Gamma.\mathrm{Comp}(m))$$
$$(\Pi \circ \Gamma).\mathrm{Dec}_k(c) = \Gamma.\mathrm{Decomp}(\Pi.\mathrm{Dec}_k(c))$$


Comp
Enc k

Technique 1: Separating secrets

Idea: use a filter to separate secrets

Suppose all secrets in a particular application have a recognizable form:

```
<form action="/money_transfer" method="post">
<input type="hidden" name="csrftoken"
       value="OWT4NmQlODE4ODRjN2Q1NTlhMmZlYWE...">
...
</form>
```

Use a filter to separate out secrets and don't compress them:

```
/value="[A-Za-z0-9]*"/
```

Filter $f : \{0,1\}^* \rightarrow \{0,1\}^* \times \{0,1\}^*$

$\mathrm{SS}_{f,\Gamma}.\mathrm{Comp}(m)$

$(pt_s, pt_{ns}) \leftarrow f(m)$
$\widetilde{pt_{ns}} \leftarrow \Gamma.\mathrm{Comp}(pt_{ns})$
return $pt_s \| \widetilde{pt_{ns}}$

$\mathrm{SS}_{f,\Gamma}.\mathrm{Decomp}(pt)$

Parse $pt_s \| \widetilde{pt_{ns}} \leftarrow pt$
$pt_{ns} \leftarrow \Gamma.\mathrm{Decomp}(\widetilde{pt_{ns}})$
$m \leftarrow f^{-1}(pt_s, pt_{ns})$
return m

CCI-security of separating secrets

Let Π be an encryption scheme.

Let Γ be a compression scheme.

Let f be a safe filter.

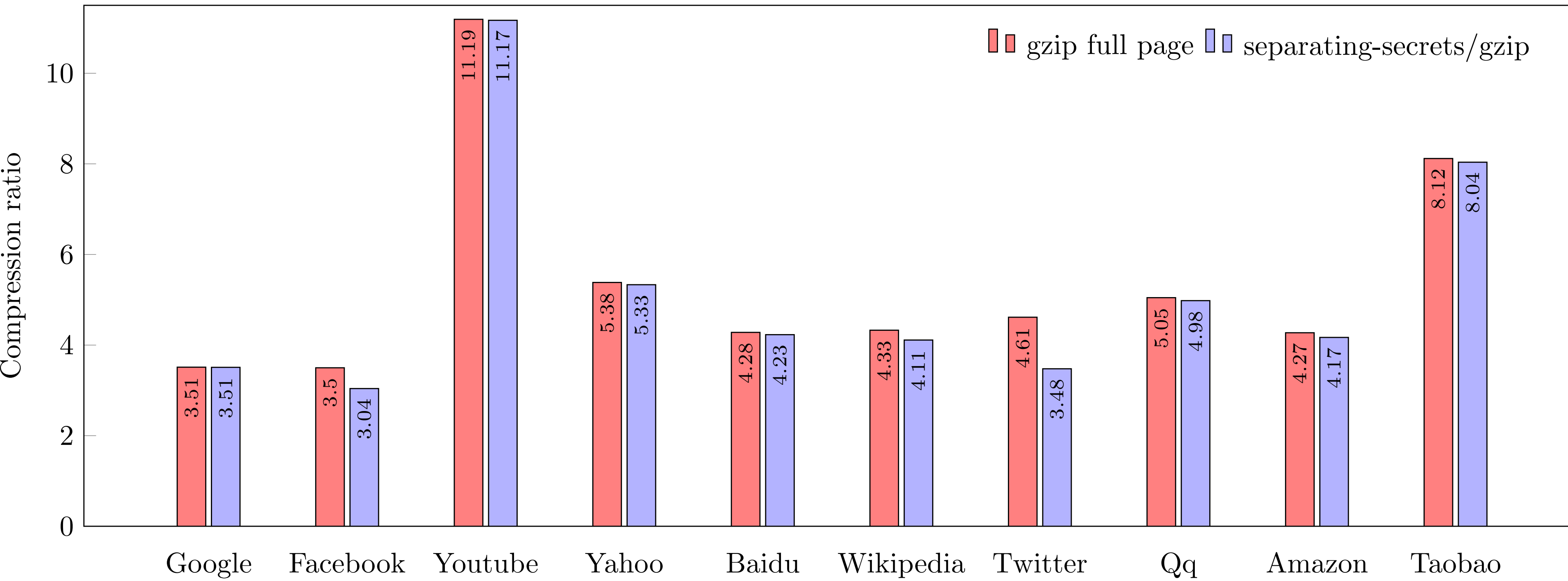
Let $\mathrm{SS}_{f,\Gamma}$ be the separating-secrets scheme using filter f and compression scheme Γ.

Then $\Pi \circ \mathrm{SS}_{f,\Gamma}$ is CCI-secure if Π is IND-CPA-secure.

$$\mathrm{Adv}^{\mathsf{CCI}}_{\Pi \circ \mathrm{SS}_{f,\Gamma}, \mathcal{CK}}(\mathcal{A}) \leq q \cdot \mathrm{Adv}^{\mathsf{IND\text{-}CPA}}_{\Pi}(\mathcal{B}^{\mathcal{A}})$$

Experimental results

`/value\s*=\s*"[A-Za-z0-9]+"|value\s*=\s*'[A-Za-z0-9]+'/`
applied to HTML/Javascript/CSS on Alexa Top 10 websites



Discussion: separating secrets

Security:

- good (CCI) security,
  provided secrets really are separated

Compression:

- very good compression assuming few secrets and efficient filter

Caveats:

- Need a good filter
  - Data marked up to clearly delineate secrets
  - Some filters separate too much and too little
    - `/value="[A-Za-z0-9]*”/`
- Application support for separating/combining secrets

Technique 2: Fixed-dictionary compression

Idea: use a fixed (non-adaptive) dictionary

- Fix a dictionary that's suitable for your typical message distribution

- To compress a message, replace words in the dictionary with their index

Basic scheme: $\mathrm{FD}_{\mathcal{D},w}$

- $\mathcal{D}$: dictionary
  - e.g., $\mathcal{D} = \texttt{cookierecoveryattack}$
- w: length of substring to try replace

$$\mathrm{FD}_{\mathcal{D},4}.\mathrm{Comp}(\text{“}\texttt{recover the cookie}\text{”}) \rightarrow \texttt{7ver\_the\_1ie}$$

CRIME-like attack against fixed dictionary

- Attacker can try prefixes/suffices that try to match the beginning/end of cookie

- $D =$ `cookierecoveryattack`
- $ck =$ `iloveyou`

- Try $m' =$ `coo` so $m' \,||\, ck =$ `cooiloveyou`
- Try $m' =$ `ook` so $m' \,||\, ck =$ `ookiloveyou`
  - This one will be compressed => CRIME attack
- Success probability falls off ~exponentially

CR-security of fixed dictionary

Let Π be an encryption scheme.

Let $\mathcal{D}$ be a dictionary of d words each of length w.

Let $\mathcal{CK} = \Omega^n$.

$$\mathrm{Adv}^{\mathsf{CR}}_{\Pi\circ\mathrm{FD}_{\mathcal{D},w,\ell}}(\mathcal{A}) \leq \mathrm{Adv}^{\mathsf{IND\text{-}CPA}}_{\Pi}(\mathcal{B}) + 2^{-\Delta}$$

where

$$\Delta \geq \left(1 - d\left(1 - \left(1 - \frac{1}{|\Omega|^w}\right)^{n-3w+1}\right)\right)$$
$$\cdot \log_2\left(|\Omega|^{n-2w} - |\Omega|^{n-2w}\cdot d\left(1 - \left(1 - \frac{1}{|\Omega|^w}\right)^{n-3w+1}\right)\right) .$$

Example parameters

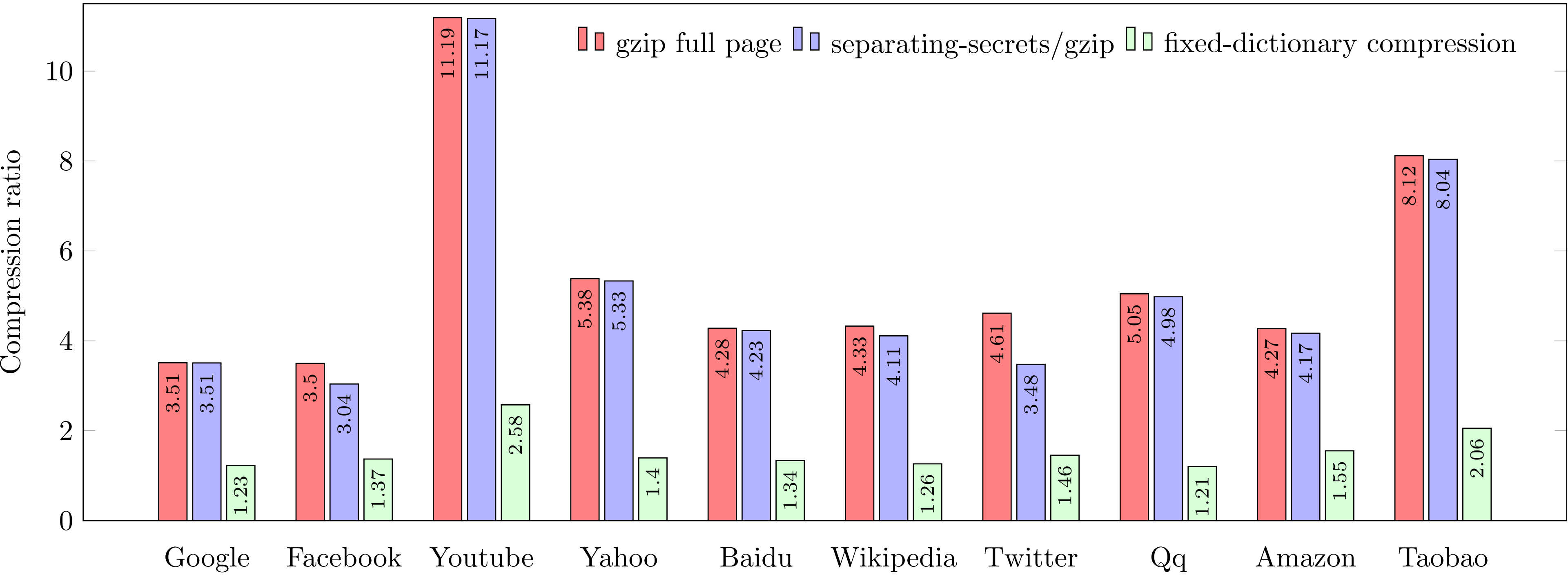
- cookies of $n = 16$ bytes
- dictionary of $d = 4000$ words each of length $w = 4$

=> $\Delta \geq 63.999695$

(compare with $8^{16} = 2^{128}$ bits of entropy)

Doubling d gives $\Delta \geq 63.999391$.

Experimental results



Discussion: fixed dictionary

Security:

- non-zero security (cookie recovery)
- not application dependent

Compression:

- poor compression

Conclusions

Recommendations from BREACH attack

1. Disabling HTTP compression
2. Separating secrets from user input
3. Randomizing secrets per request
4. Masking secrets (effectively randomizing by XORing with a random nonce)
5. Length hiding (by adding a random number of bytes to the responses)
6. Rate-limiting the requests
7. Use non-adaptive compression algorithm

Summary of results

Security Definitions

- Cookie recovery (CR)
- Random cookie indistinguishability (RCI)
- Chosen cookie indistinguishability (CCI)

- Relations and separations
  - CCI => RCI => CR
  - ER-IND-CPA => IND-CPA => CCI

Techniques

Separating secrets:

- CCI-secure with a good filter

Fixed-dictionary:

- CR-secure with high-entropy secrets

Unsatisfying answers

- Separating secrets technique requires a good data-specific filter and application changes to be secure
- Fixed dictionary compression is more reliably secure but much poorer compression

- *Unavoidable*: Basic combination of compression and encryption will always leak some information about the plaintext

Surely we can do something better?

Something interesting: HPACK

http://http2.github.io/http2-spec/compression.html

- Header compression for HTTP/2
- Every header and every component of every header is compressed in its own context
  - Implementations can disable compression for "valuable" headers
- Uses a pre-established static dictionary + a dynamic dictionary
- Body still compressed all-at-once using gzip

- Merits more investigation

Protecting encrypted cookies from compression side-channel attacks

Janaka Alawatugoda, Douglas Stebila (QUT), Colin Boyd (NTNU) ▪ FC 2015 ▪ eprint 2014/724

Security Definitions

- Cookie recovery (CR)
- Random cookie indistinguishability (RCI)
- Chosen cookie indistinguishability (CCI)
- Separations and relations

Techniques

Separating secrets:

- CCI-secure with a good filter

Fixed-dictionary:

- CR-secure with high-entropy secrets

Future Directions

- Analysis of HPACK
- Where else is compression used?