



# CTX: Eliminating BREACH with Context Hiding

Dimitris Karakostas  
Aggelos Kiayias  
Eva Sarafianou  
Dionysis Zindros





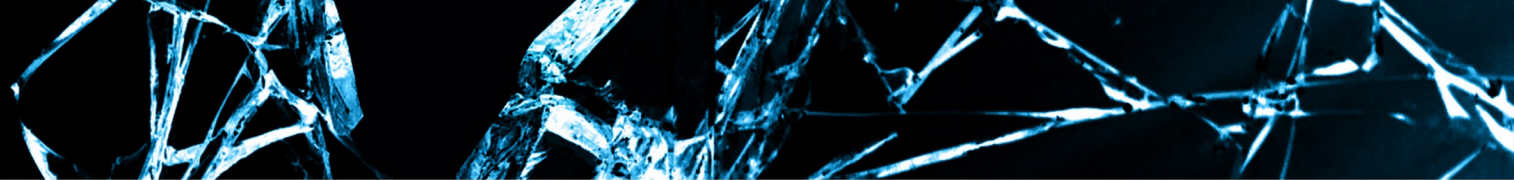
# Who are we?

**Dimitris Karakostas, Eva Sarafianou, Dionysis Zindros**

Researchers at Security & Cryptography lab  
University of Athens, Greece

**Aggelos Kiayias**

Chair in Cyber Security and Privacy  
University of Edinburgh, Scotland



# HTTPS is broken

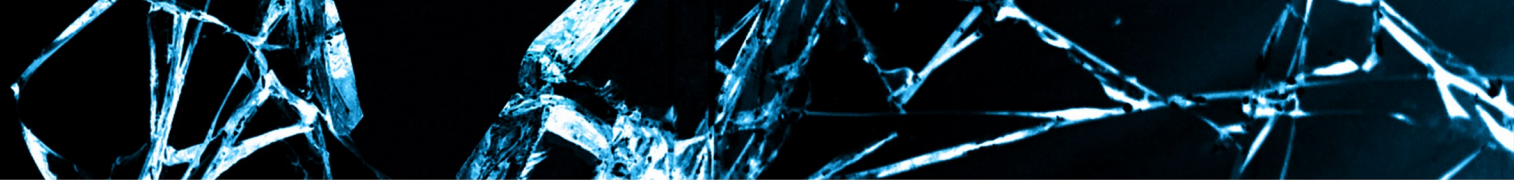
- BREACH broke HTTPS + RC4 in 2013
- People upgraded to AES – thought they were safe
- Rupture attacked HTTPS with block ciphers

## Today...

- We show a generic defense for compression side-channel attacks
- Best balance between compression and security
- We launch an open source implementation of the defense for popular web frameworks

# Overview

- Introduction
  - History
  - Attack vectors
- The CTX defense
  - Origins, Secrets, Cross compression
  - Permutations
  - CTX architecture
- Release
- Future work



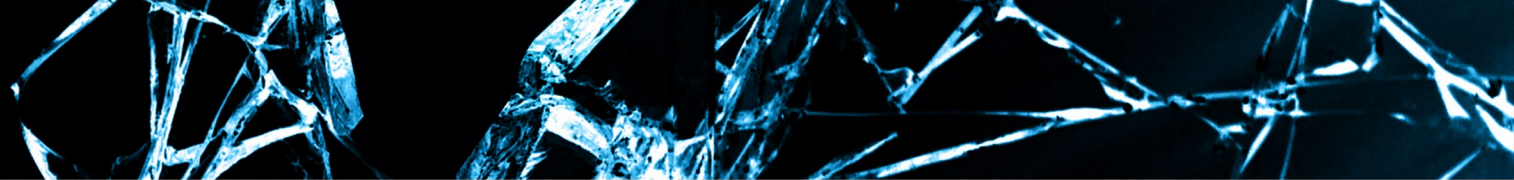
# CRIME, 2012

- Targets HTTPS requests
- Side-channel compression attacks against TLS first-time successful
- Takes advantage of the characteristics of the DEFLATE algorithm
- Hinted at attacking responses
- Mitigated by disabling compression at the TLS level

# TIME, 2013

- Exploits compression on HTTP responses
- Exploits compression by measuring time transmission
- No need for permanent Man-in-the-Middle agents





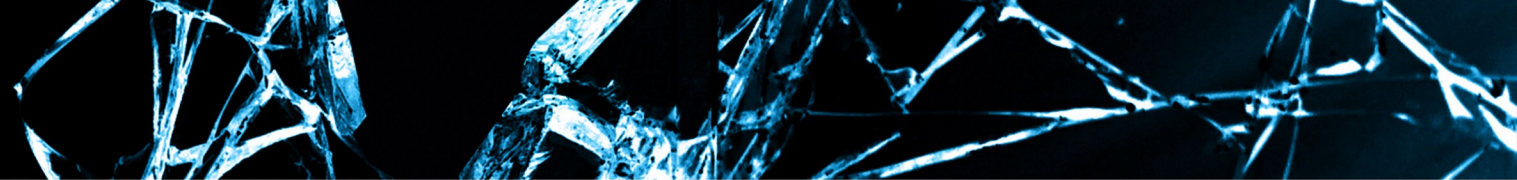
# BREACH, 2013

- Exploits compression on HTTP response body
- Attacks stream ciphers
- Adds methods for bypassing compression noise

# RC4 insecurity, 2015

- RC4 is considered insecure
- Most websites use block ciphers
- AES is the industry standard





# Rupture, 2016

- Exploits compression on HTTP responses
- Performs statistical analysis
- Bypasses noise/length hiding
- Attacks block ciphers, eg AES
- Automates the attack process
- Production code

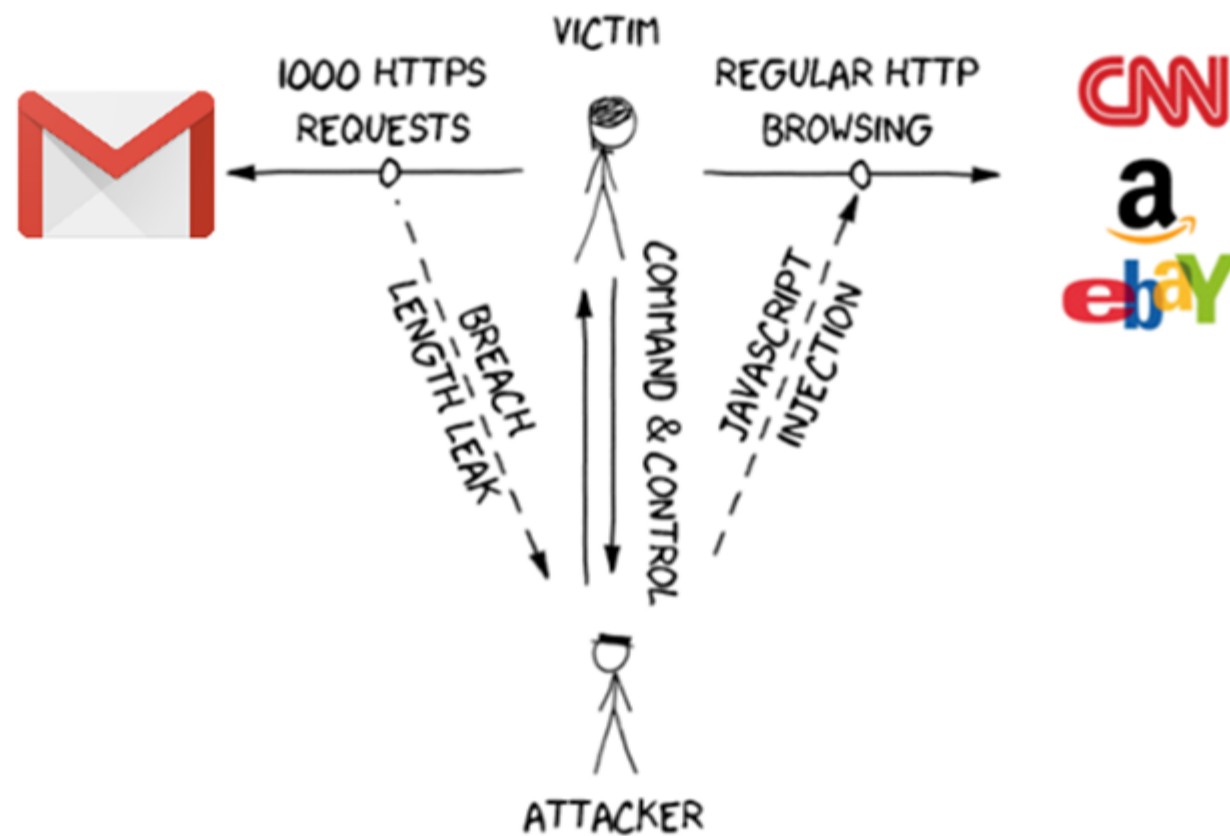
# HEIST, 2016

- No need for Man-in-the-Middle agents to perform BREACH
- Abuses the way responses are sent at the TCP level

# Attack methodology

- Compression is better across same content
  - Example: “test\_test” compresses better than “test\_rand”
- Method
  - Target an HTTPS website
  - Find a web page that:
    - Allows parameter *reflection*
    - Contains a *secret*
  - Issue requests with different reflections using the victim’s cookies
  - Measure the responses’ lengths
  - Decrypt the secret using statistical analysis





- Attacker guesses part of secret
- Uses it in reflection
- Compressed/encrypted response is **shorter if right!**

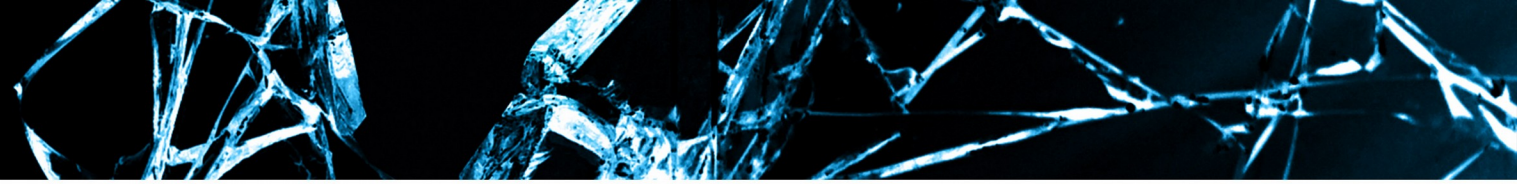
```
base href="https://mail.google.com/mail/u/0/x/puqq7ui43zaf-/" />
value="?&at=AF6bupMJX-9CU4zxp362SDbN49o45nMjSg&s=q" />
type="hidden" name="nredir" value="?&q=blackhatblackhat&am
/><input type="hidden" name="search" value="query" /><div
class="noMatches">No results for: AF6bupMJX-9CU4 </div><scrip
type="text/javascript">
var token="AF6bupMJX-9CU4zxp362SDbN49o45nMjSg";var
searchPageLinks=document.getElementsByClassName("searchPageLin
for(i=0;i<searchPageLinks.length;i++)searchPageLinks[i].onclick
```

**Reflection**

**Secret**

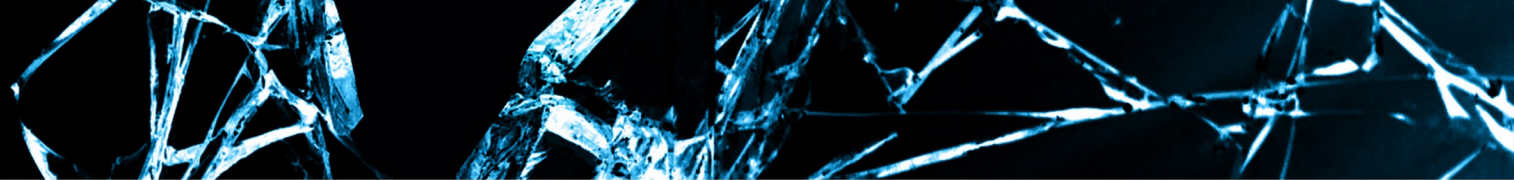
# The CTX defense





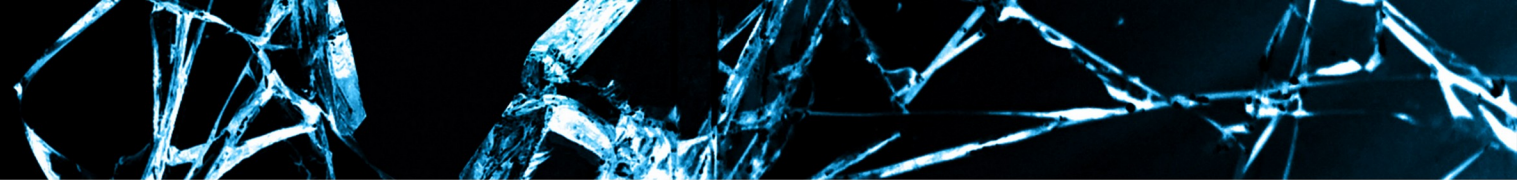
# CTX, Context Transformation Extension


Context hiding in a **per-origin** manner  
to separate **secrets** and avoid **cross-compression**





# Origin


- Party that generated the secret
  - Web application
  - User
- Secrets of the same origin → Cross-compression
- Secrets of different origin → Separate compression














☐





More ▾

1–15 of 15










☐




**Eva Sarafianou**

» **New otr fingerprint** - Check my website for my new otr fingerprint

12:17 am

☐




**Dimitris Karakostas**

» **Important information** - This is not an email.

12:16 am

☐




**Eva Sarafianou**

» **Thesis draft** - Find the first draft of my thesis attached

12:15 am

☐

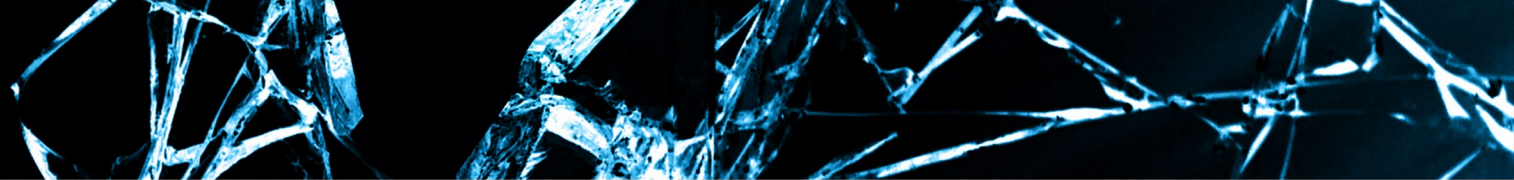


**Dimitris Karakostas**

» **Paper info** - This is a confidential mail.

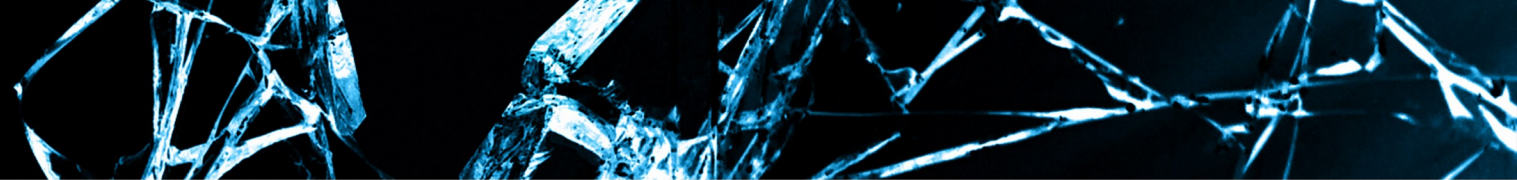
12:13 am








# Secret


- Parts of the response
  - CSRF tokens
  - Private messages
  - E-mails
  - Financial data
- Any piece of information which is only accessible when logged in














☐





More ▾

1–15 of 15










☐




**Eva Sarafianou**

» New otr fingerprint - Check my website for my new otr fingerprint

12:17 am

☐




**Dimitris Karakostas**

» Important information - This is not an email.

12:16 am

☐




**Eva Sarafianou**

» Thesis draft - Find the first draft of my thesis attached

12:15 am

☐




**Dimitris Karakostas**


» Paper info - This is a confidential mail.


12:13 am


OK to compress  
together


















☐ 









More 

1–15 of 15

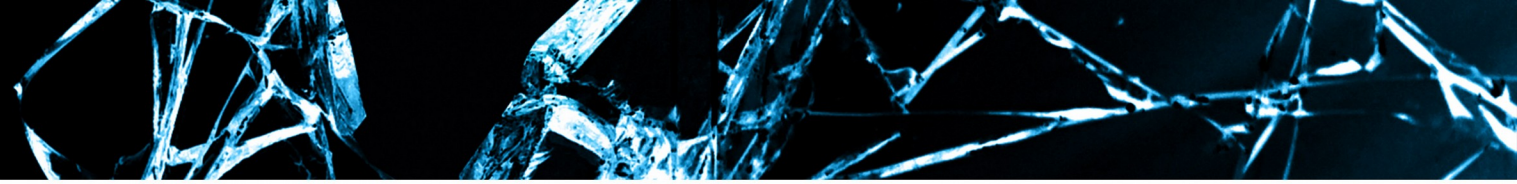
 

<input type="checkbox"/>		Eva Sarafianou	» New otr fingerprint - Check my website for my new otr fingerprint	12:17 am
<input type="checkbox"/>		Dimitris Karakostas	» Important information - This is not an email.	12:16 am
<input type="checkbox"/>		Eva Sarafianou	» Thesis draft - Find the first draft of my thesis attached	12:15 am
<input type="checkbox"/>		Dimitris Karakostas	» Paper info - This is a confidential mail.	12:13 am

NOT OK to compress together!

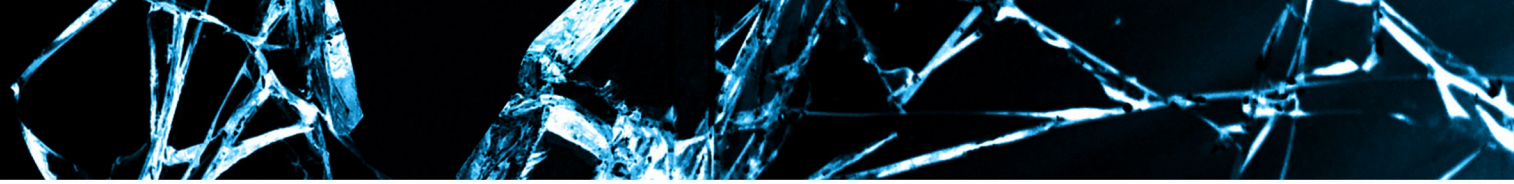


# Cross-compression

- Cross-compression between “a”, “b” → Presence of “a” affects compression of “b”
- Example:
  - LZ77 compression
  - Plaintext: a + b
  - a = “secret1”, b = “secret2”
  - Cross-compression:
    - $C(a) = \text{“secret1”}$ ,  $C(b) = (7, 6) + \text{“2”}$
  - Separate compression:
    - $C(a) = \text{“secret1”}$ ,  $C(b) = \text{“secret2”}$

# How can we protect secrets?

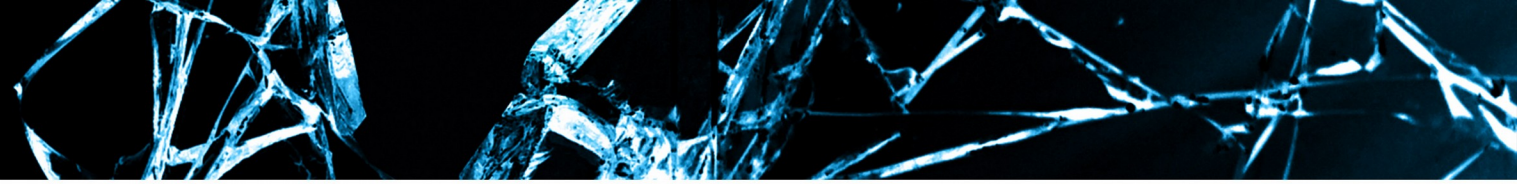
- Disable compression ✗
  - Unacceptable performance penalty
- Change the compression function ✗
  - All good compression functions are vulnerable
- Modify the web server compression module ✗
  - Requires changing both the web server & application
  - Hard to achieve good compression rate
- Hide length with random padding (TLS 1.3) ✗
  - Susceptible alignment + statistical analysis (Rupture)
- Change the response plaintext ✓



# CTX, Context Transformation Extension

- Protects HTTPS responses
- Runs at the application layer
- Is opt-in
- Balances between performance and security
  - Slight compression size increase
  - Small time performance overhead
  - Fully prevents complete plaintext recovery
  - Successful defense for all known compression attacks (TIME, CRIME, BREACH etc)





# CTX, Context Transformation Extension

Application developer must do the following:

- Import ctx library server-side ( Django, Flask, Node.js ... )
- Import ctx library client-side ( `<script src="ctx.js"></script>` )
- Select sensitive secrets
- Define origin for each secret

```
<body>
  <table>
    <tr><td>From</td><td>Body</td></tr>

    {% for email in emails: %}
      <tr>
        <td> {{ email.sender }} </td>
        <td> {{ ctx_protect(email.body, email.sender) }} </td>
      </tr>
    {% endfor %}
  </table>

  {{ ctx_permutations() }}
  <script src="ctx.js"></script>
</body>
```

```
<body>
  <table>
    <tr><td>From</td><td>Body</td></tr>

    {% for email in emails %}
      <tr>
        <td> {{ email.sender }} </td>
        <td> {{ ctx_protect(email.body, email.sender) }} </td>
      </tr>
    {% endfor %}
  </table>

  {{ ctx_permutations() }}
  <script src="ctx.js"></script>
</body>
```

↑ Secret      ↑ Origin

```
<body>
  <table>
    <tr><td>From</td><td>Body</td></tr>

    <tr>
      <td> dimkarakostas@gmail.com </td>
      <td> Hello Dionyziz, Black Hat Asia 2017 application details. </td>
    </tr>

    <tr>
      <td> eva.sarafianou@gmail.com </td>
      <td> My master thesis draft attached. </td>
    </tr>

    <tr>
      <td> dimkarakostas@gmail.com </td>
      <td> Question on Kademlia internals. </td>
    </tr>
  </table>
</body>
```

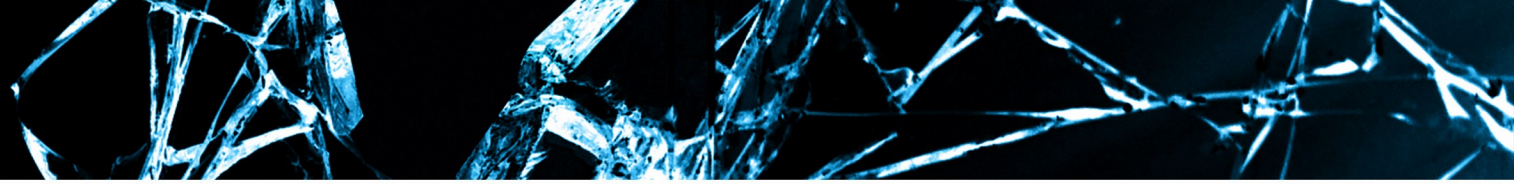


```
<body>
  <table>
    <tr><td>From</td><td>Body</td></tr>

    <tr>
      <td> dimkarakostas@gmail.com </td>
      <td> <div data-ctx-origin='0'>fh%60%606%21-
%286Qkt%28ti%21%5D%60%22%237%21f%22U%21v%5E%28%22%21nX%2B%2C%21%22//%60%28%23%22U%286Q%21FhU%22%28%60%5EL</div> </td>
    </tr>

    <tr>
      <td> eva.sarafianou@gmail.com </td>
      <td> <div data-ctx-origin='1'>K%3D%3A%29%26%21D_%7C%3ADf_%21%0C%21%3A%7B%7C%26jD%3A%26DD%26hf_%7BP</div> </td>
      <td> </td>
    </tr>

    <tr>
      <td> dimkarakostas@gmail.com </td>
      <td> <div data-ctx-origin='0'>G%29h%5EU%286Q%216Q%21K%22Fh1%60%28%22%21%28QUh9Q%22%60%5EL</div> </td>
    </tr>
  </table>
</body>
```

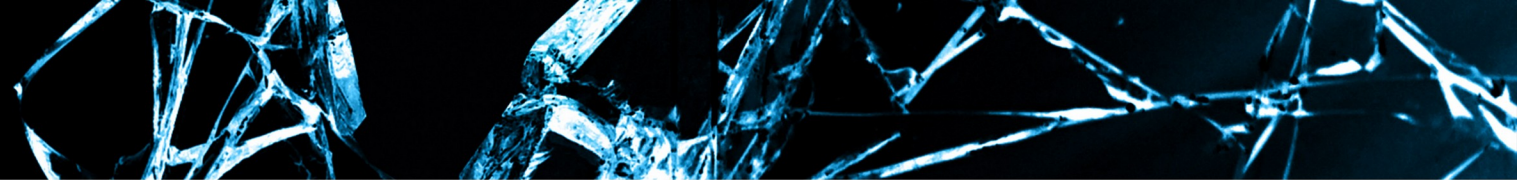


# Permutations

- Define secret alphabet
  - Contains all possible characters in the secret
    - e.g. ASCII, UTF-8
- Pseudo-random permutation of the secret alphabet for each origin
- Fisher-Yates shuffle algorithm
- Permute secrets using the origin's permutation
- TLS encryption and network transmission of the permuted secret
- Apply inverse permutation → Decode the secret

Secret	Origin	Permuted secret
secret1	origin1	)o5eoc8
secret2	origin1	)o5eock
secret3	origin2	heb^eV#

Origin	Permutation
origin1	s → )    e → o c → 5    r → e t → c    1 → 8 2 → k    3 → # (...)
origin2	s → h    e → e c → b    r → ^ t → V    1 → g 2 → !    3 → # (...)



# Attack mitigated

- New per-origin permutations per HTTP response
- Multiple responses contain differently permuted secrets
- Permutations cannot be statistically predicted



# Performance experiments

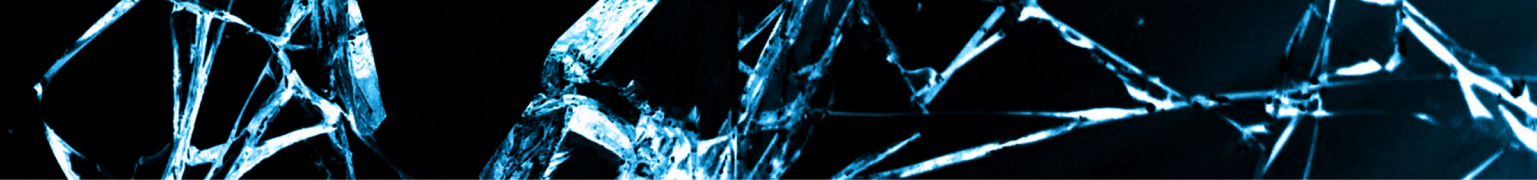
- We test size/time performance under CTX
- Test web page:
  - 650KB (e.g. YouTube timeline)
  - 50 origins
  - 1% secrets in the response equally distributed in origins
  - 1 secret position per origin

# Performance experiments

- Results:
  - Disable total compression:
    - 1,100% size overhead
    - Few *seconds* time delay during transmission
  - Masking secrets:
    - 21% size overhead
  - CTX:
    - 5% size overhead ~ 7KB
    - 4ms time delay

# Performance experiments

- Origins ↑ → Performance ↓
- Total secrets ↑ → Performance ↓
- Secrets per origin ↑ → Performance ↑
- Total response ↑ → Performance ↑

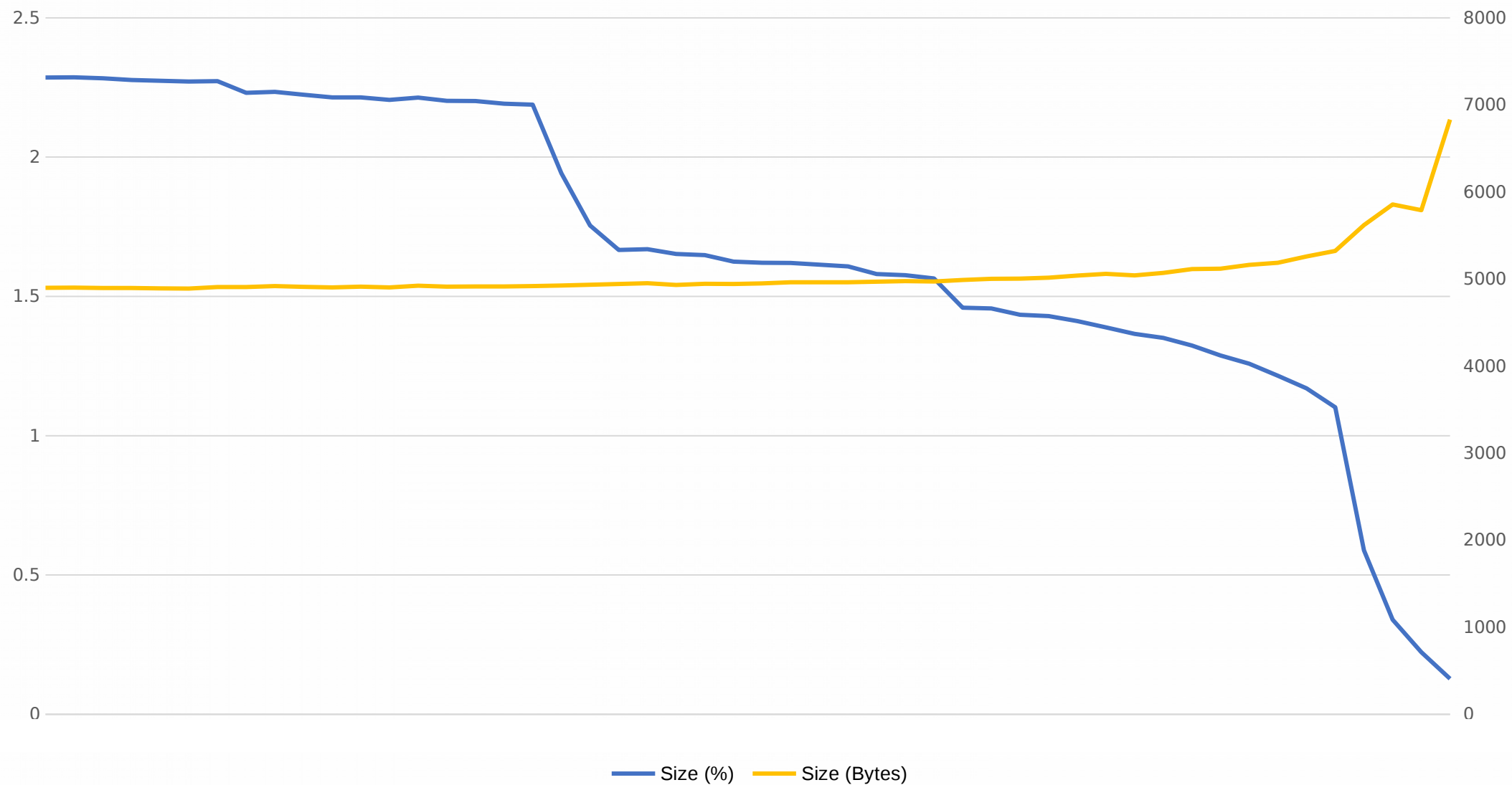


# Total response performance

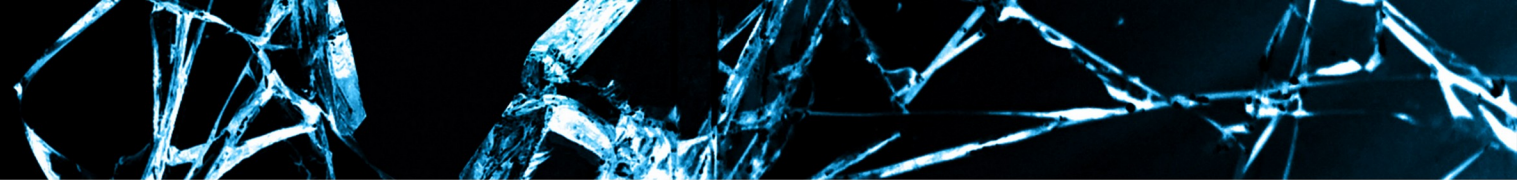
- Bigger response:
  - Similar byte size overhead
  - Better percentage size overhead



## Total response

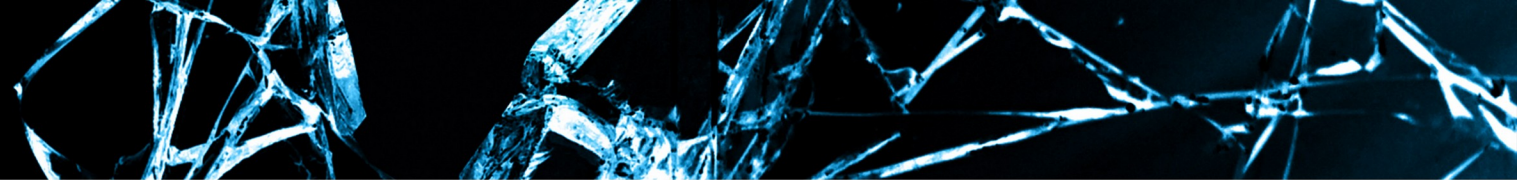


# CTX Architecture



# CTX Architecture

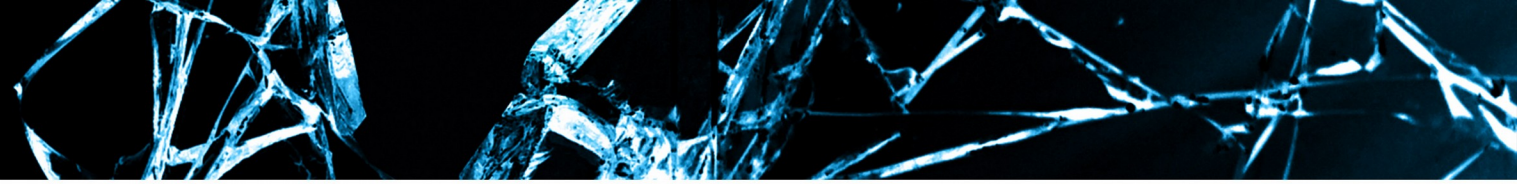
- Server
  - Parses HTML for ctx-protect div tags
  - Creates permutation for every new origin
  - Permutes secrets in a per-origin manner
  - Includes a JSON file with all permutations
  - Sends response containing permuted secrets and permutations



# Client

- Parses the HTML for data-ctx-origin div tags
- Parses the JSON and collects each origin's permutation
- Applies reverse permutation on each secret





# Today, we defend BREACH attacks

- Today in Black Hat Europe 2016, we launch CTX for popular web frameworks
  - Python: Django, Flask
  - Node.js: Express [express-Handlebars, pug (jade), EJS], Koa [koa-pug]
- Open source - MIT licensed

<https://github.com/dimkarakostas/ctx>

<https://ctxdefense.com>

# Future Work

- Implement CTX for other languages/web frameworks
- Extend CTX for other encoding standards
- Implement CTX for API web frameworks

# Key Takeaways

1. HTTPS + gzip = broken
2. CTX provides full security
3. **Add CTX protection to your web applications**

# Thank you! Questions?

<https://dimkarakostas.com>

DF46 7AFF 3398 BB31 CEA7 1E77 F896 1969 A339 D2E9

<http://www.kiayias.com>

E5F2 7045 437B 168B 39AD 1BFA C876 8019 6DBB 04E0

<https://esarafianou.github.io>

2FA9 7528 9554 F1EB F5F8 675B E371 5849 8CD0 92EE

<https://dionyziz.com>

45DC 00AE FDDF 5D5C B988 EC86 2DA4 50F3 AFB0 46C7