Proxying is Enough

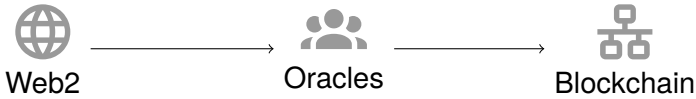
Security of Proxying in TLS Oracles and AEAD Context Unforgeability

Zhongtang Luo[1], Yanxue Jia[1], Yaobin Shen[2], Aniket Kate[13]

[1]Purdue University, [2]Xiamen University, [3]Supra Research

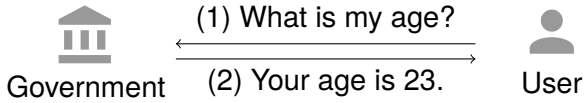
SBC 2024
Last Compiled:
July 25, 2024

Oracle



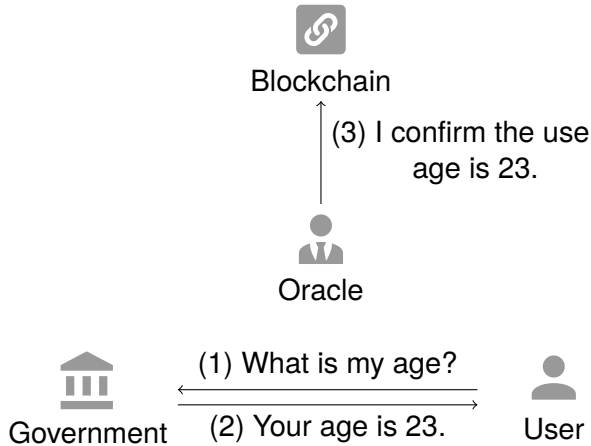
Oracles pull in information from Web2.

Oracle

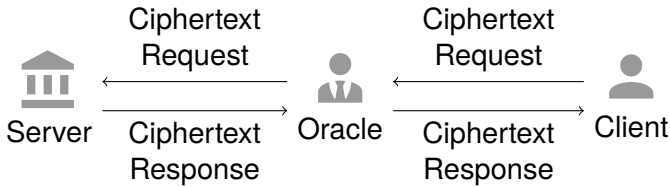


How can we pull in more information?

TLS Oracle


Blockchain
(3) I confirm the user's age is 23.
Oracle
Government
(1) What is my age?
(2) Your age is 23.
User

Proxy-Based TLS Oracle



The user reveals the needed part of the plaintext at the end (with some proof).

Big question: Is it secure?

Proxy-Based TLS Oracle

- Conjectured **insecure** since DECO in 2020

Proxy-Based TLS Oracle

- Conjectured **insecure** since DECO in 2020
- Key commitment attack: The ciphertext may decrypt to a different plaintext with a different key.

Proxy-Based TLS Oracle

- Conjectured **insecure** since DECO in 2020
- Key commitment attack: The ciphertext may decrypt to a different plaintext with a different key.
- A whole plethora of work on ensuring key commitment:
  - *DECO: Liberating Web Data Using Decentralized Oracles for TLS*
  - *DIDO: Data Provenance from Restricted TLS 1.3 Websites*
  - *Janus: Fast Privacy-Preserving Data Provenance for TLS*
  - *Lightweight Authentication of Web Data via Garble-Then-Prove*
  - *ORIGO: Proving Provenance of Sensitive Data with Constant Communication*
  - ...

But is it really insecure?

Take 1: HTTPS

- Popular fix for key commitment: Padding[1]

[1] https://eprint.iacr.org/2020/1456

Take 1: HTTPS

- Popular fix for key commitment: Padding[1]
- Now let us look at HTTPS...

[1] https://eprint.iacr.org/2020/1456

Take 1: HTTPS

- Popular fix for key commitment: Padding[1]
- Now let us look at HTTPS...

```
HTTP/1.1 200 OK
Date: Wed, 24 Jul 2024 23:41:36 GMT
Expires: -1
Cache-Control: private, max-age=0
Content-Type: text/html; charset=UTF-8
...
```

https://google.com

```
HTTP/1.1 200 OK
date: Wed, 24 Jul 2024 23:47:49 GMT
perf: 7402827104
expiry: Tue, 31 Mar 1981 05:00:00 GMT
pragma: no-cache
...
```

https://twitter.com

[1] https://eprint.iacr.org/2020/1456

Take 1: HTTPS

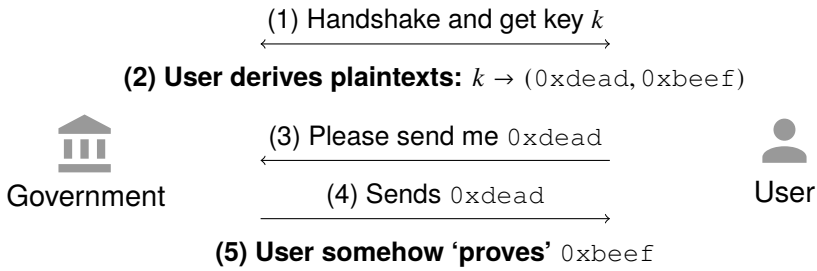
- It turns out that HTTPS is (variably) padded!
- Specifics vary, but most start with status code and date
  - also recommended by RFC 7231

Take 1: HTTPS

- It turns out that HTTPS is (variably) padded!
- Specifics vary, but most start with status code and date
  - also recommended by RFC 7231
- **We proved that proxy-based TLS is secure for HTTPS.**
  - **Covers almost all websites!**

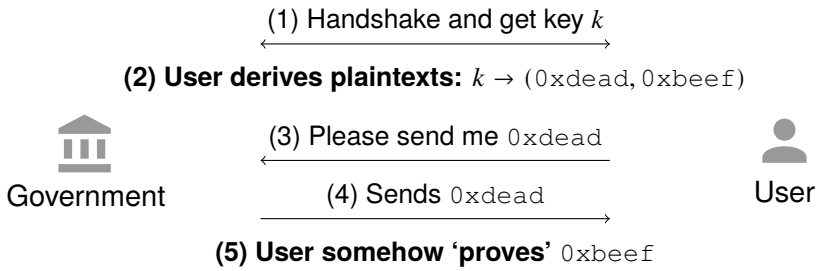
Take 2: Non-HTTPS

- Attacks incoming...



Take 2: Non-HTTPS

- Attacks incoming...



How likely will this attack happen?

Take 2: Non-HTTPS

Malleable

Account balance
Bank statement
...

Insecure

Fixed

Account number
Age
...

Insecure?

Take 2: Non-HTTPS

- For fixed data, we need only a weaker key commitment property for the cipher suite.
  - We define as **context unforgeability (CFY)**.
  - Informally: For fixed plaintext, hard to find another plaintext that matches the ciphertext
  - Like second-preimage resistance in hash functions

Take 2: Non-HTTPS

AES-GCM	Chacha20-Poly1305
AES is a block cipher (reversible).	Chacha20 is based on PRF (not reversible).
Not secure under CFY	Secure under CFY
Cannot be used in non-HTTPS scenarios	**Can** be used in non-HTTPS scenarios with fixed data
	

Takeaways

Proxy-Based TLS Oracles

HTTPS

Secure!

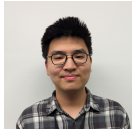
99% of use case

Non-HTTPS

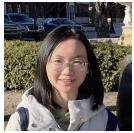
Secure?

Make sure data is fixed
Use Chacha20-Poly1305

Authors



Zhongtang Luo



Yanxue Jia

Yaobin Shen

Aniket Kate

Links

Paper

https://eprint.iacr.org/2024/733

Slides

https://zhtluo.com/paper/Proxying_is_Enough__Security_of_Proxying_in_TLS_Oracles_and_AEAD_Context_Unforgeability_Slides.pdf