The 9 Lives of Bleichenbacher's CAT: New Cache ATtacks on TLS Implementations

Eyal Ronen, Robert Gillham, Daniel Genkin, Adi Shamir, David Wong and Yuval Yarom

מכון ויצמן למדע
WEIZMANN INSTITUTE OF SCIENCE

UNIVERSITY OF MICHIGAN

TEL AVIV UNIVERSITY אוניברסיטת תל אביב

DATA 61

THE UNIVERSITY of ADELAIDE

Transport Layer Security (TLS)

- The most widely used cryptographic protocol
- Provides communication security (https, VPN, etc.)
  - TLS handshake is used for authentication and secure key exchange
  - TLS Record layer protects the communication
  - Allows for cryptographic agility using different cipher suites

RSA Key Exchange in TLS

- Uses the PKCS #1 v1.5 padding scheme

RSA Key Exchange in TLS

- Uses the PKCS #1 v1.5 padding scheme
- Once the most popular TLS key exchange option

RSA Key Exchange in TLS

- Uses the PKCS #1 v1.5 padding scheme
- Once the most popular TLS key exchange option
- Long history of practical **implementation** attacks*

RSA Key Exchange in TLS

- Uses the PKCS #1 v1.5 padding scheme
- Once the most popular TLS key exchange option
- Long history of practical **implementation** attacks*
- No forward secrecy

RSA Key Exchange in TLS

- Uses the PKCS #1 v1.5 padding scheme
- Once the most popular TLS key exchange option
- Long history of practical **implementation** attacks*
- No forward secrecy
- Still widely used (Dec 2018)
  - ~6% by Mozilla's Telemetry and ICSI Certificate Notary

RSA Key Exchange in TLS

- Uses the PKCS #1 v1.5 padding scheme
- Once the most popular TLS key exchange option
- Long history of practical **implementation** attacks*
- No forward secrecy
- Still widely used (Dec 2018)
  - ~6% by Mozilla's Telemetry and ICSI Certificate Notary
  - Better alternatives now available (e.g. Ephemeral ECDH)

RSA Key Exchange in TLS

- Uses the PKCS #1 v1.5 padding scheme
- Once the most popular TLS key exchange option
- Long history of practical **implementation** attacks*
- No forward secrecy
- Still widely used (Dec 2018)
    - ~6% by Mozilla's Telemetry and ICSI Certificate Notary
    - Better alternatives now available (e.g. Ephemeral ECDH)
    - Supported for backwards compatibility

9 lives of Bleichenbacher’s CAT

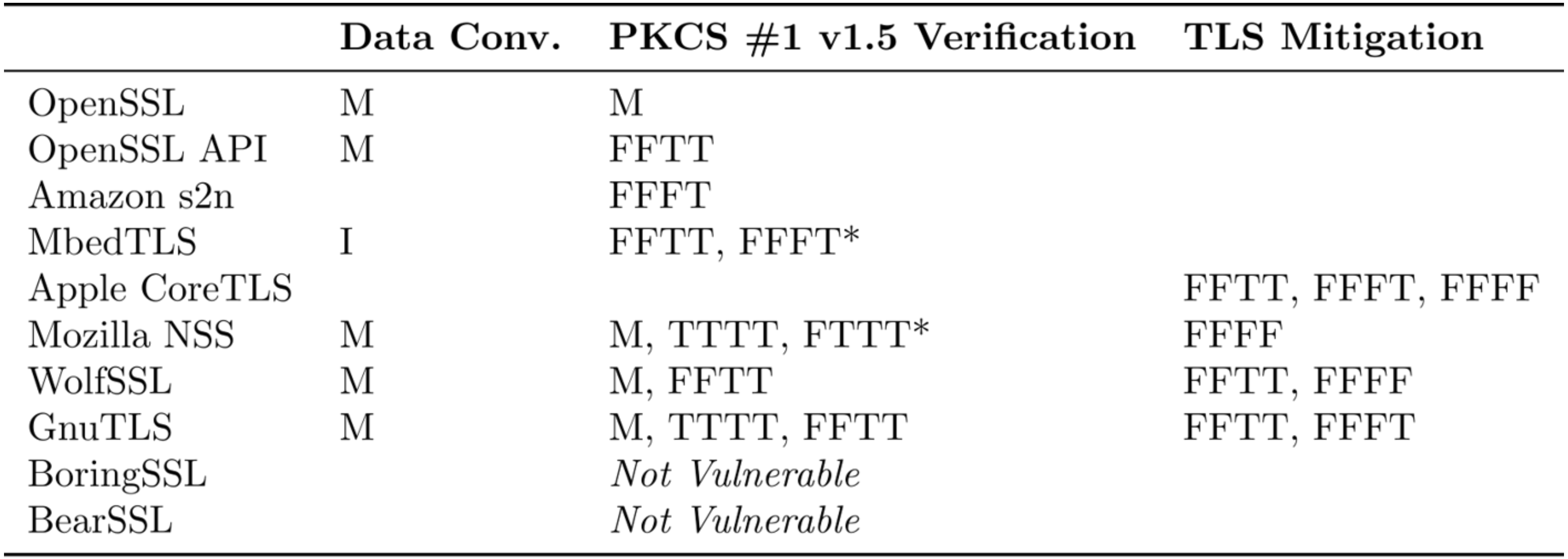
- We tested the latest version of 9 different TLS implementations

9 lives of Bleichenbacher’s CAT

- We tested the latest version of 9 different TLS implementations
- 7 found vulnerable to new cache based RSA padding attacks

9 lives of Bleichenbacher’s CAT

- We tested the latest version of 9 different TLS implementations
- 7 found vulnerable to new cache based RSA padding attacks
    - Multiple vulnerabilities in different layers of the protocol



	Data Conv.	PKCS #1 v1.5 Verification	TLS Mitigation
OpenSSL	M	M	
OpenSSL API	M	FFTT	
Amazon s2n		FFFT	
MbedTLS	I	FFTT, FFFT*	
Apple CoreTLS			FFTT, FFFT, FFFF
Mozilla NSS	M	M, TTTT, FTTT*	FFFF
WolfSSL	M	M, FFTT	FFTT, FFFF
GnuTLS	M	M, TTTT, FFTT	FFTT, FFFT
BoringSSL		*Not Vulnerable*	
BearSSL		*Not Vulnerable*	

9 lives of Bleichenbacher's CAT

- We broke 6% of the Internet, so what?
    - Only old clients use RSA KX

9 lives of Bleichenbacher’s CAT

- We broke 6% of the Internet, so what?
  - Only old clients use RSA KX
- We show the feasibility of MiTM downgrade attack

9 lives of Bleichenbacher’s CAT

- We broke 6% of the Internet, so what?
  - Only old clients use RSA KX
- We show the feasibility of MiTM downgrade attack
  - Cause modern client and server to use RSA KX

9 lives of Bleichenbacher's CAT

- We broke 6% of the Internet, so what?
    - Only old clients use RSA KX
- We show the feasibility of MiTM downgrade attack
    - Cause modern client and server to use RSA KX
    - Novel parallelization technique for RSA padding oracle attacks

9 lives of Bleichenbacher's CAT

- We broke 6% of the Internet, so what?
  - Only old clients use RSA KX
- We show the feasibility of MiTM downgrade attack
  - Cause modern client and server to use RSA KX
  - Novel parallelization technique for RSA padding oracle attacks
  - Break 100% of the connections to servers that use vulnerable implementations

9 lives of Bleichenbacher's CAT

- We broke 6% of the Internet, so what?
  - Only old clients use RSA KX
- We show the feasibility of MiTM downgrade attack
  - Cause modern client and server to use RSA KX
  - Novel parallelization technique for RSA padding oracle attacks
  - Break 100% of the connections to servers that use vulnerable implementations
  - Works also if client doesn't support RSA KX

RSA Encryption

$$N = p \cdot q \qquad\qquad (p, q) \text{ are primes}$$
$$d \cdot e = 1 \mod \phi(N)$$
$$c = m^e \mod N$$
$$m = c^d \mod N$$

RSA Encryption

$$N = p \cdot q \qquad (p, q) \text{ are primes}$$
$$d \cdot e = 1 \mod \phi(N)$$
$$c = m^e \mod N$$
$$m = c^d \mod N$$

- Nice math, but how can we use it on real data?

RSA Encryption

$$N = p \cdot q \qquad (p, q) \text{ are primes}$$
$$d \cdot e = 1 \mod \phi(N)$$
$$c = m^e \mod N$$
$$m = c^d \mod N$$

- Nice math, but how can we use it on real data?
  - There are several real world problems

Why do we need padding

- Assume $e = 3$, $m = 1000$, $N \sim 2^{2048}$

Why do we need padding

- Assume $e = 3$, $m = 1000$, $N \sim 2^{2048}$
  - $m^e < N$, logarithm over the reals is easy
  - m should be larger

Why do we need padding

- Assume $e = 3$, $m = 1000$, $N \sim 2^{2048}$
  - $m^e < N$, logarithm over the reals is easy
  - m should be larger
- Assume encryption of Yes/No – value 0 or 1

Why do we need padding

- Assume $e = 3$, $m = 1000$, $N \sim 2^{2048}$
  - $m^e < N$, logarithm over the reals is easy
  - m should be larger
- Assume encryption of Yes/No – value 0 or 1
  - Vulnerable to dictionary attack
  - Easy to detect repetitions
  - m should be random

PKCS #1 v1.5 padding scheme

- Used to pad and encrypt the plaintext
  - Pads the plaintext to the RSA public key length
  - Adds randomization

PKCS #1 v1.5 padding scheme

- Used to pad and encrypt the plaintext
  - Pads the plaintext to the RSA public key length
  - Adds randomization
- Example for RSA key exchange in TLS 1.2

0x0002	[non-zero padding]	0x00	[48 bytes of premaster secret]

PKCS #1 v1.5 padding scheme

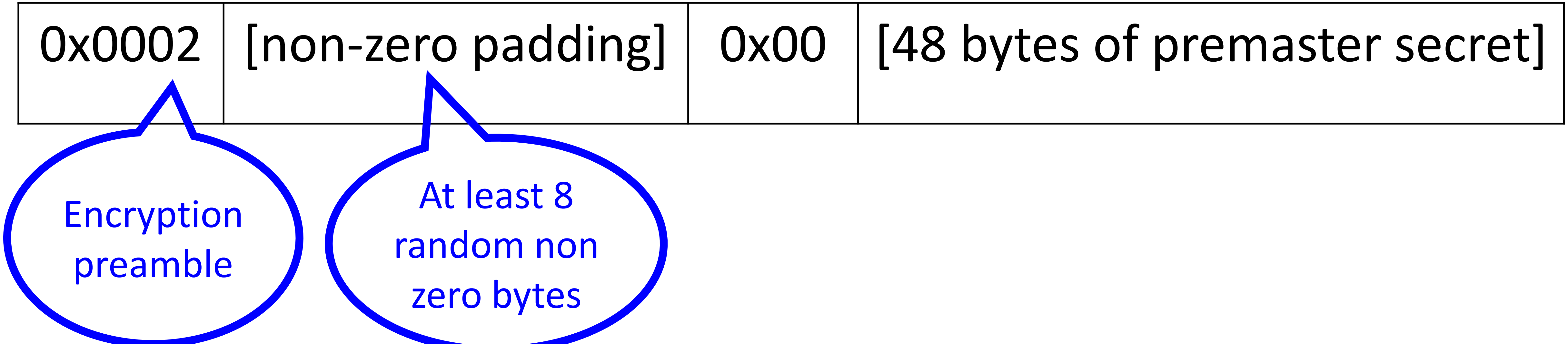
- Used to pad and encrypt the plaintext
  - Pads the plaintext to the RSA public key length
  - Adds randomization
- Example for RSA key exchange in TLS 1.2

0x0002	[non-zero padding]	0x00	[48 bytes of premaster secret]

Encryption preamble

PKCS #1 v1.5 padding scheme

- Used to pad and encrypt the plaintext
  - Pads the plaintext to the RSA public key length
  - Adds randomization
- Example for RSA key exchange in TLS 1.2



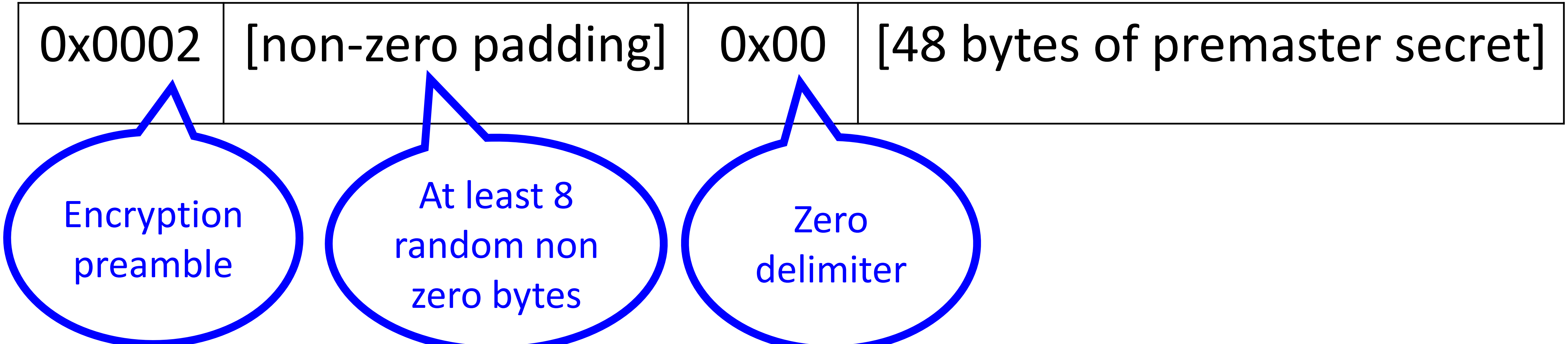
0x0002	[non-zero padding]	0x00	[48 bytes of premaster secret]

Encryption preamble

At least 8 random non zero bytes

PKCS #1 v1.5 padding scheme

- Used to pad and encrypt the plaintext
  - Pads the plaintext to the RSA public key length
  - Adds randomization
- Example for RSA key exchange in TLS 1.2



0x0002	[non-zero padding]	0x00	[48 bytes of premaster secret]

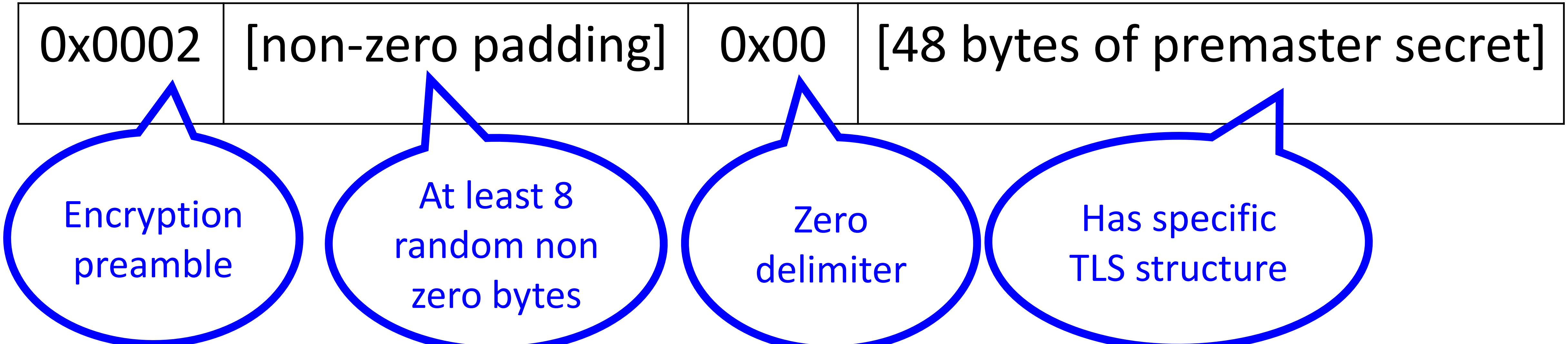
Encryption preamble

At least 8 random non zero bytes

Zero delimiter

PKCS #1 v1.5 padding scheme

- Used to pad and encrypt the plaintext
  - Pads the plaintext to the RSA public key length
  - Adds randomization
- Example for RSA key exchange in TLS 1.2



0x0002	[non-zero padding]	0x00	[48 bytes of premaster secret]

Encryption preamble

At least 8 random non zero bytes

Zero delimiter

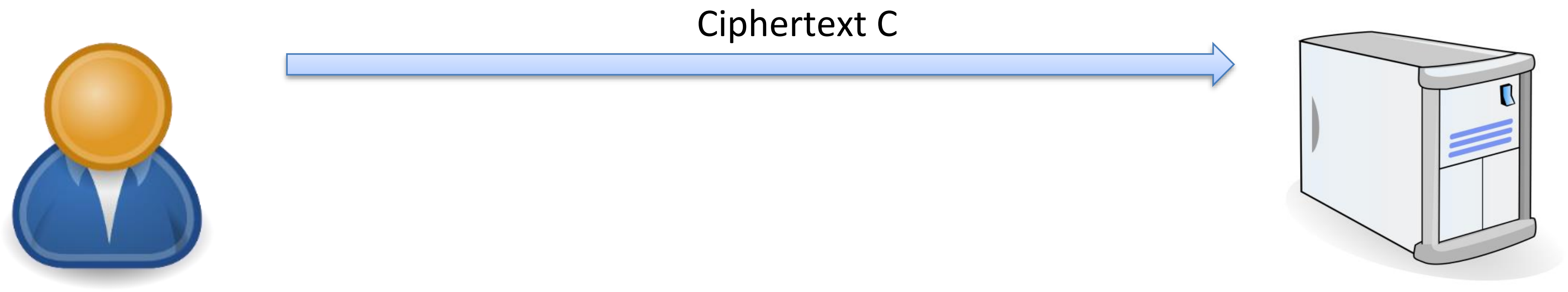
Has specific TLS structure

Bleichenbacher’s Attack

- 1998: Adaptive chosen-ciphertext attack
- Exploits strict RSA PKCS#1 v1.5 padding validation

Bleichenbacher’s Attack

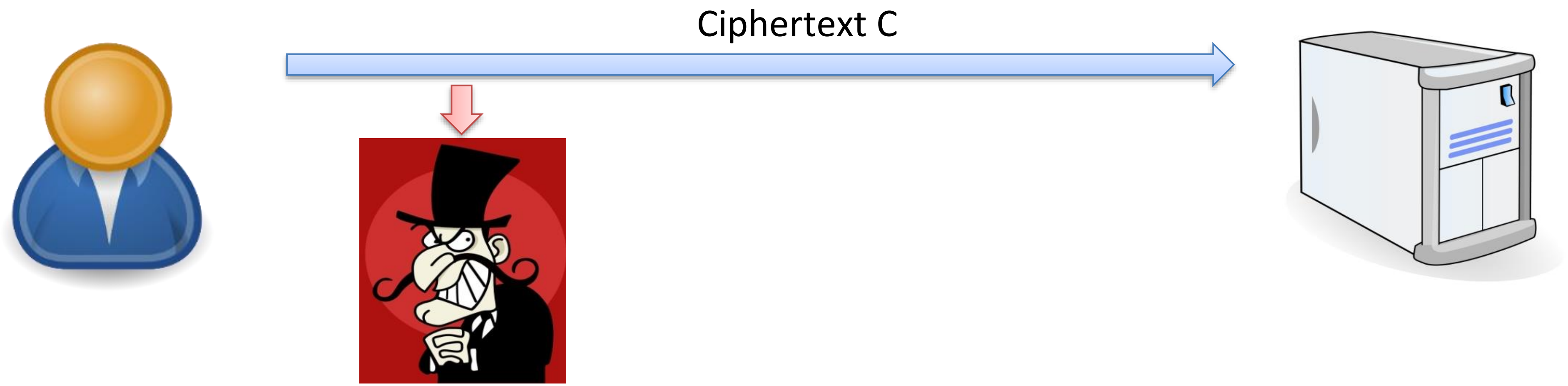
- 1998: Adaptive chosen-ciphertext attack
- Exploits strict RSA PKCS#1 v1.5 padding validation



Ciphertext C

Bleichenbacher’s Attack

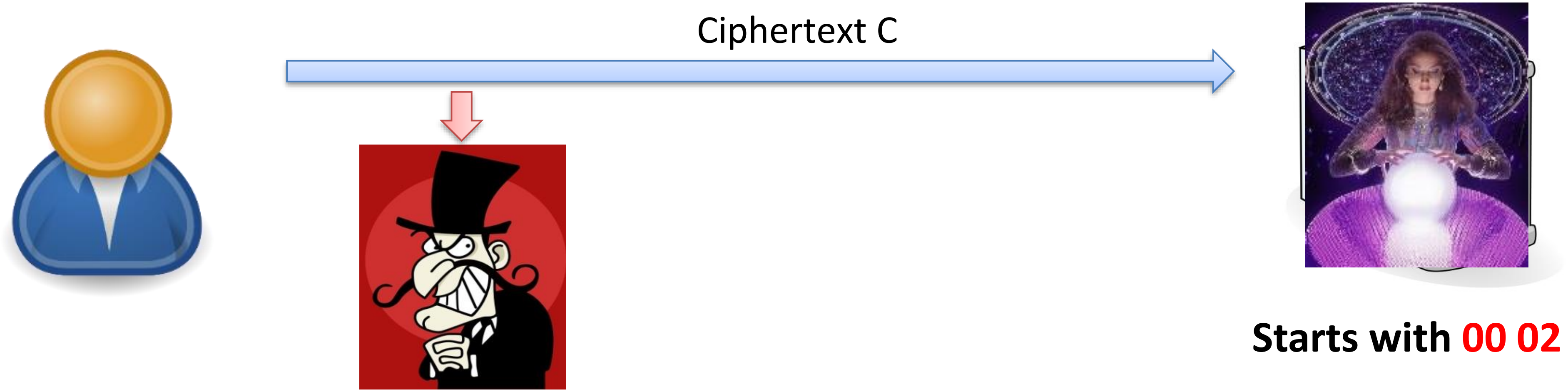
- 1998: Adaptive chosen-ciphertext attack
- Exploits strict RSA PKCS#1 v1.5 padding validation



Ciphertext C

Bleichenbacher’s Attack

- 1998: Adaptive chosen-ciphertext attack
- Exploits strict RSA PKCS#1 v1.5 padding validation

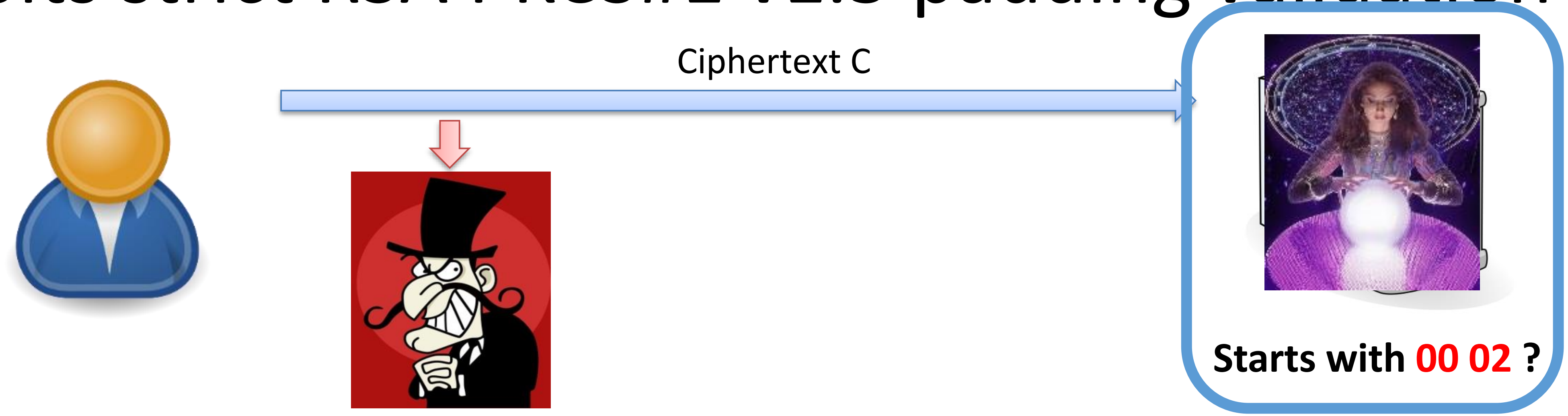


Ciphertext C

Starts with 00 02 ?

Bleichenbacher's Attack

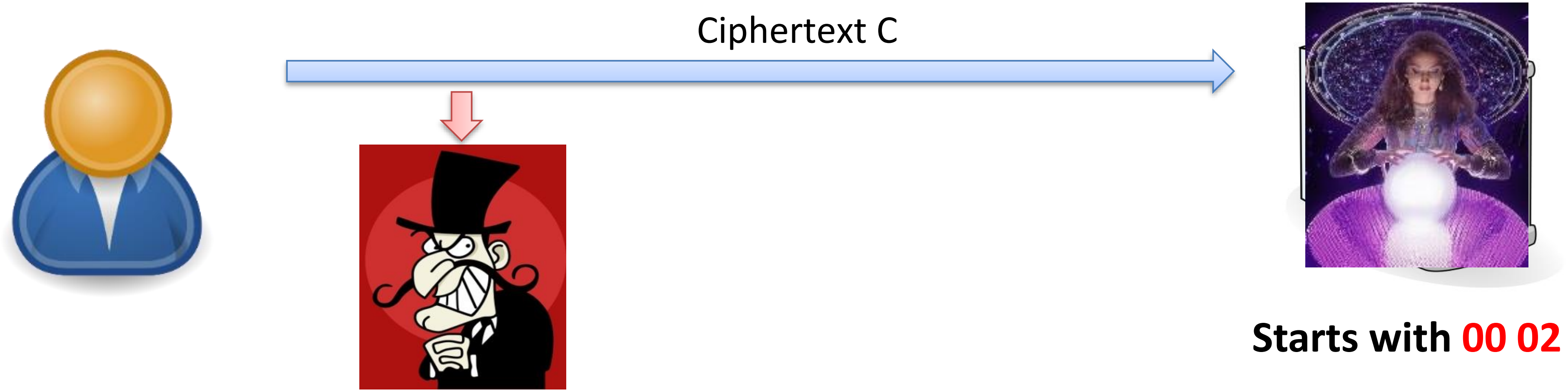
- 1998: Adaptive chosen-ciphertext attack
- Exploits strict RSA PKCS#1 v1.5 padding validation



Requires Side Channel

Bleichenbacher’s Attack

- 1998: Adaptive chosen-ciphertext attack
- Exploits strict RSA PKCS#1 v1.5 padding validation

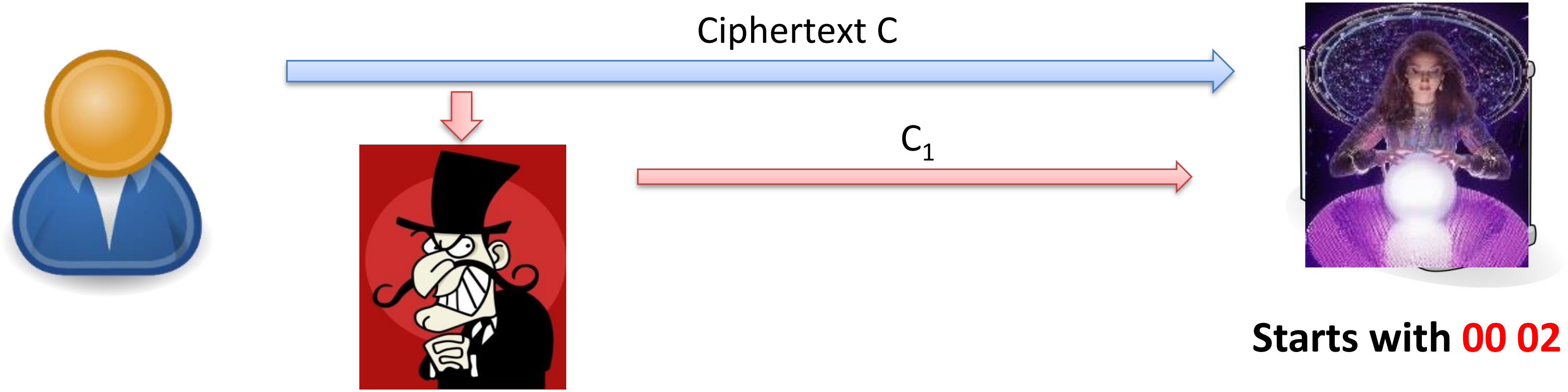


Ciphertext C

Starts with 00 02 ?

Bleichenbacher’s Attack

- 1998: Adaptive chosen-ciphertext attack
- Exploits strict RSA PKCS#1 v1.5 padding validation



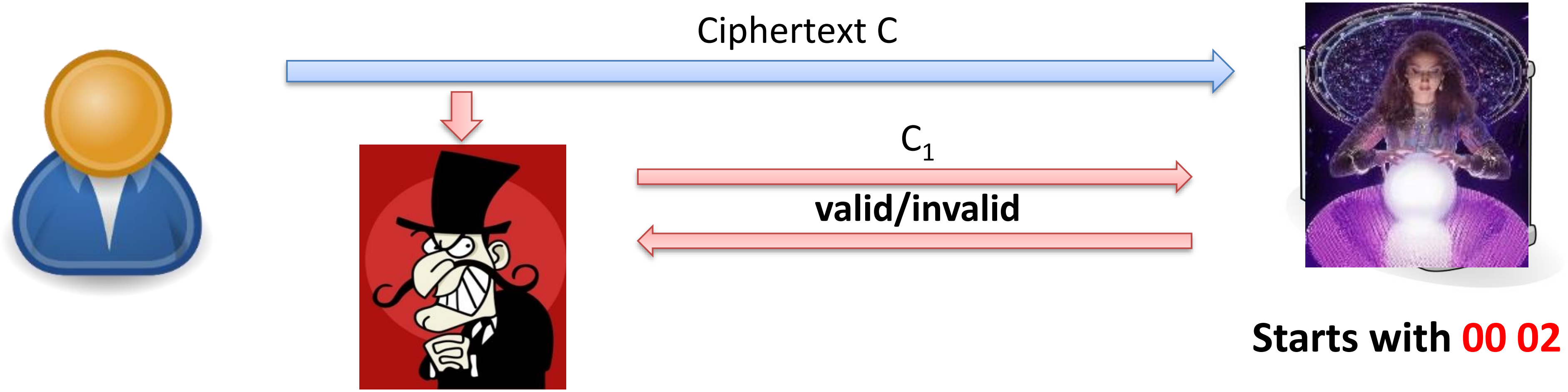
Ciphertext C

C_1

Starts with <span>00 02</span> ?

Bleichenbacher’s Attack

- 1998: Adaptive chosen-ciphertext attack
- Exploits strict RSA PKCS#1 v1.5 padding validation



Ciphertext C

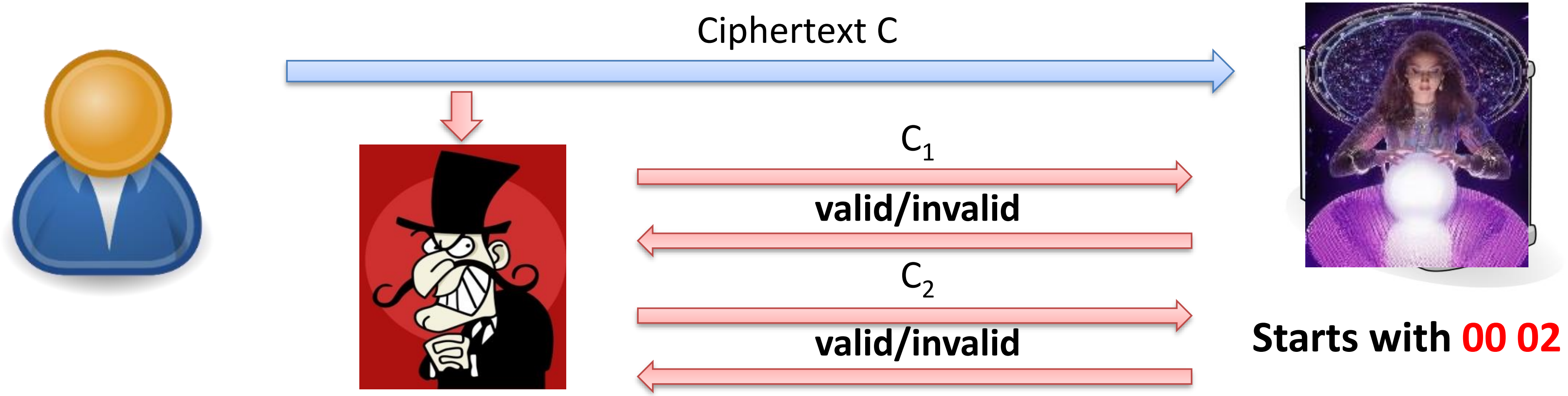
C_1

valid/invalid

Starts with 00 02 ?

Bleichenbacher's Attack

- 1998: Adaptive chosen-ciphertext attack
- Exploits strict RSA PKCS#1 v1.5 padding validation



Ciphertext C

C_1

valid/invalid

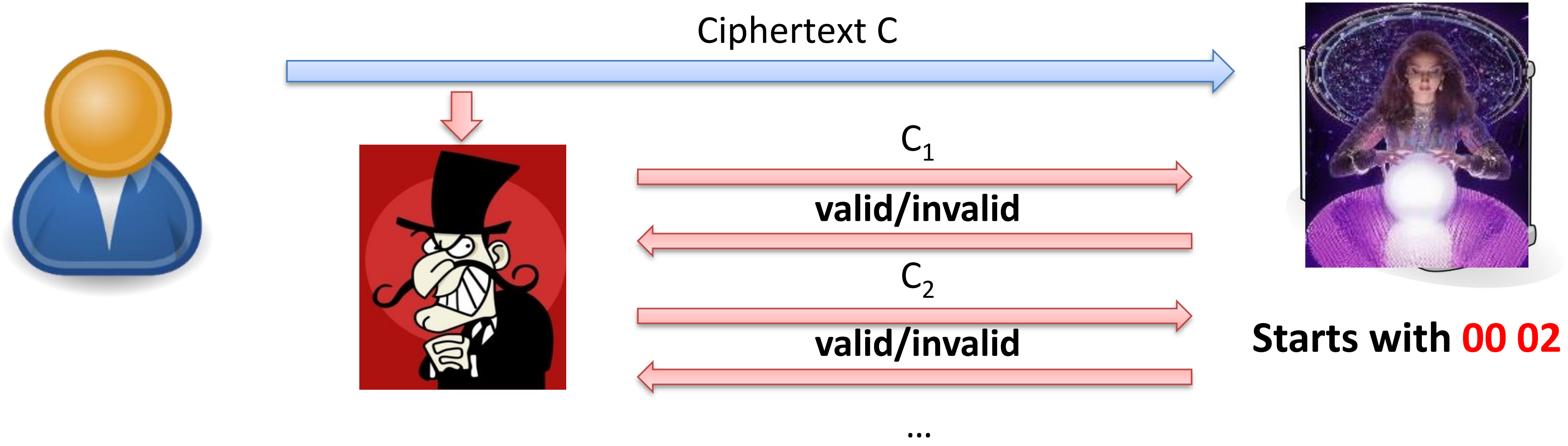
C_2

valid/invalid

Starts with 00 02 ?

Bleichenbacher’s Attack

- 1998: Adaptive chosen-ciphertext attack
- Exploits strict RSA PKCS#1 v1.5 padding validation



Ciphertext C

C_1

valid/invalid

C_2

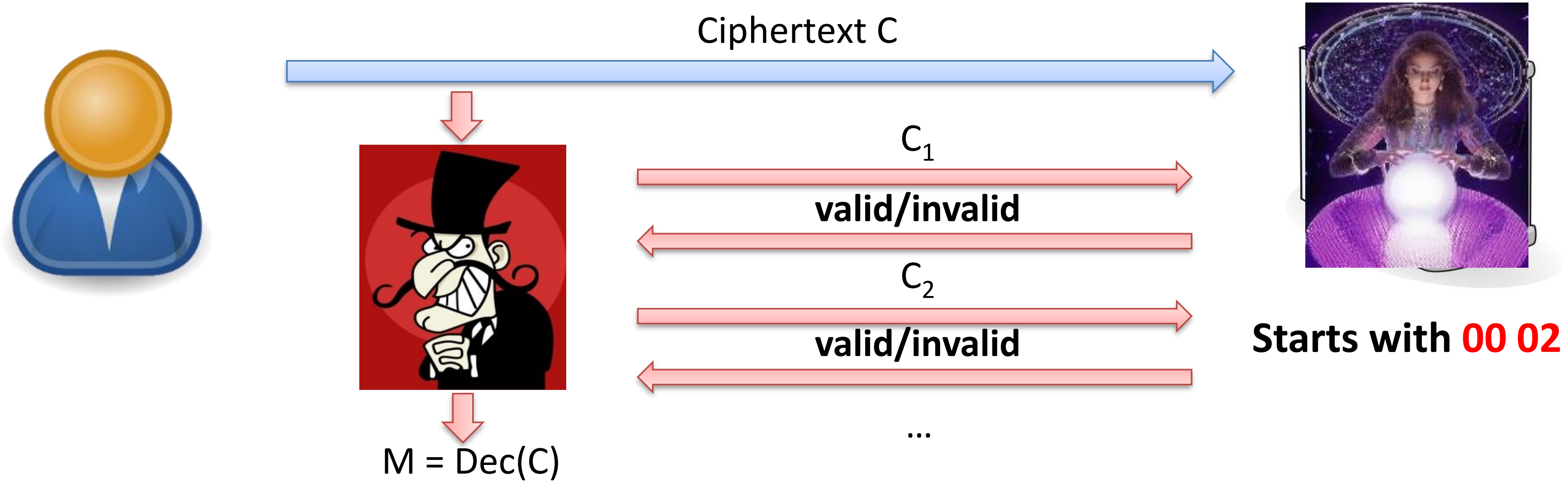
valid/invalid

…

Starts with 00 02 ?

Bleichenbacher's Attack

- 1998: Adaptive chosen-ciphertext attack
- Exploits strict RSA PKCS#1 v1.5 padding validation



Ciphertext C

C_1

valid/invalid

C_2

valid/invalid

...

M = Dec(C)

Starts with 00 02 ?

ME WANT COOKIE!

- Session cookies give access to the users’ data
    - Are sent in the beginning of each TLS connection

ME WANT COOKIE!

- Session cookies give access to the users' data
  - Are sent in the beginning of each TLS connection
- Attack scenario for RSA KX:

ME WANT COOKIE!

- Session cookies give access to the users' data
    - Are sent in the beginning of each TLS connection
- Attack scenario for RSA KX:
    - Sniff TLS handshake and first message

ME WANT COOKIE!

- Session cookies give access to the users' data
  - Are sent in the beginning of each TLS connection
- Attack scenario for RSA KX:
  - Sniff TLS handshake and first message
  - Use Bleich. to decrypt premaster secret

ME WANT COOKIE!

- Session cookies give access to the users' data
  - Are sent in the beginning of each TLS connection
- Attack scenario for RSA KX:
  - Sniff TLS handshake and first message
  - Use Bleich. to decrypt premaster secret
  - Decrypt first message

ME WANT COOKIE!

- Session cookies give access to the users’ data
  - Are sent in the beginning of each TLS connection
- Attack scenario for RSA KX:
  - Sniff TLS handshake and first message
  - Use Bleich. to decrypt premaster secret
  - Decrypt first message
  - COOKIE!

Attack Scenario RSA KX:
Sniff + Cache timing side channel

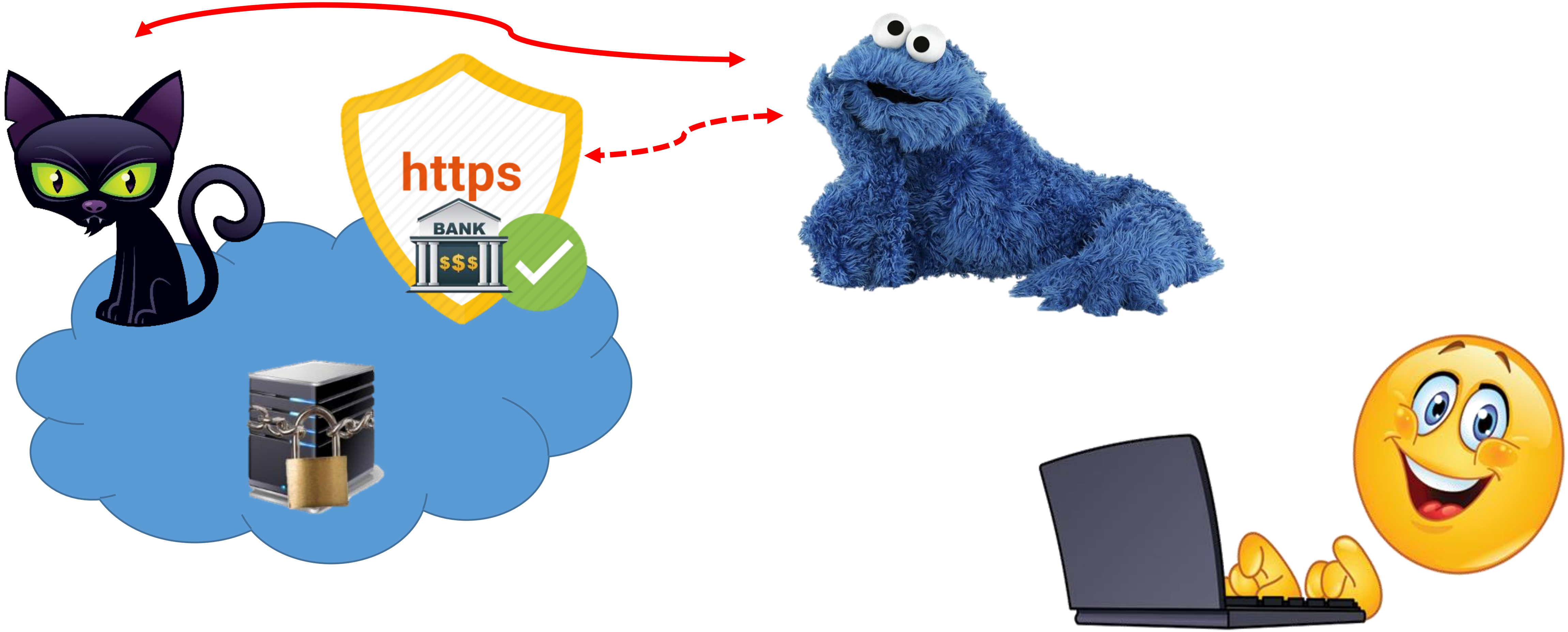
Attack Scenario RSA KX:
Sniff + Cache timing side channel

Attack Scenario RSA KX:
Sniff + Cache timing side channel

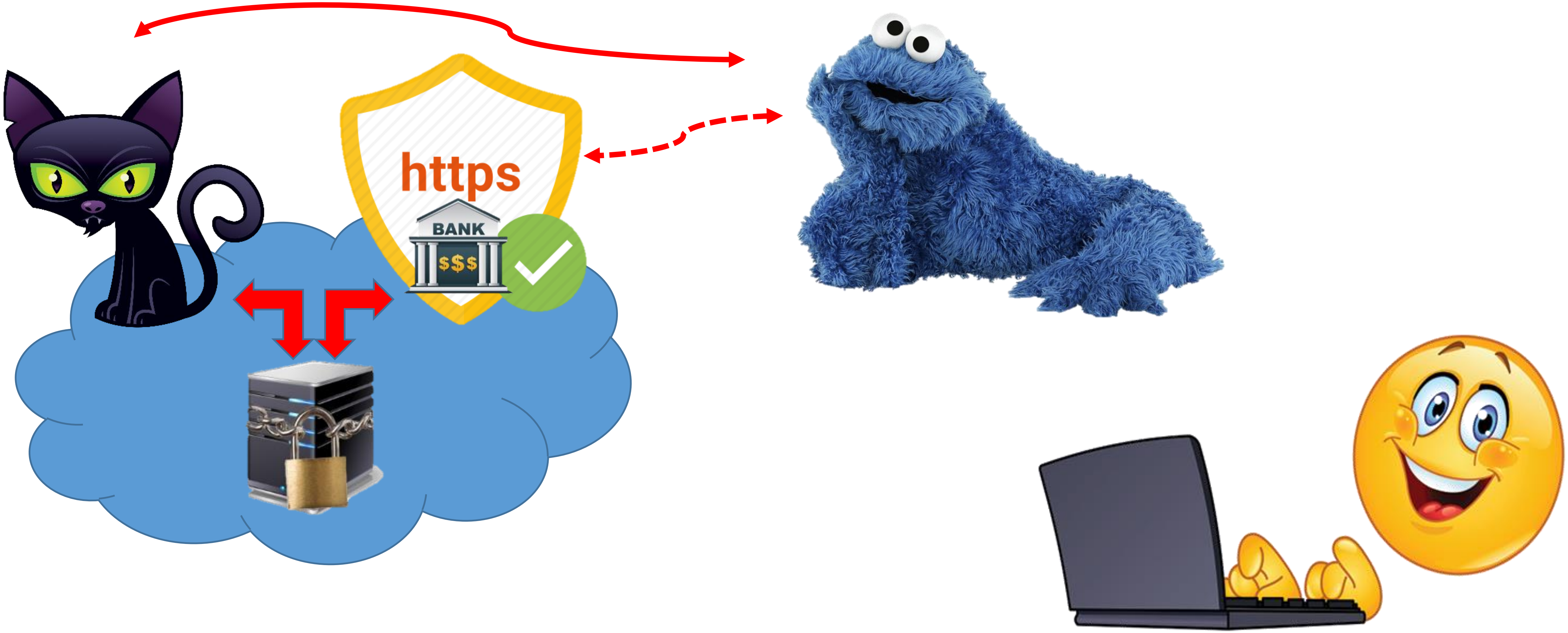
Attack Scenario RSA KX:
Sniff + Cache timing side channel

Attack Scenario RSA KX:
Sniff + Cache timing side channel

Attack Scenario RSA KX:
Sniff + Cache timing side channel



Attack Scenario RSA KX:
Sniff + Cache timing side channel



Attack Scenario RSA KX:
Sniff + Cache timing side channel

ME WANT COOKIE! ALL COOKIES!

- Only 6% of connections use RSA KX

ME WANT COOKIE! ALL COOKIES!

- Only 6% of connections use RSA KX
- Use RSA KX vulnerability for downgrade attack

ME WANT COOKIE! ALL COOKIES!

- Only 6% of connections use RSA KX
- Use RSA KX vulnerability for downgrade attack
  - Only requires server support for RSA KX

ME WANT COOKIE! ALL COOKIES!

- Only 6% of connections use RSA KX
- Use RSA KX vulnerability for downgrade attack
  - Only requires server support for RSA KX
  - Works also on TLS 1.3 [JSS 15]

ME WANT COOKIE! ALL COOKIES!

- Only 6% of connections use RSA KX
- Use RSA KX vulnerability for downgrade attack
  - Only requires server support for RSA KX
  - Works also on TLS 1.3 [JSS 15]
  - Require active MiTM attack

ME WANT COOKIE! ALL COOKIES!

- Only 6% of connections use RSA KX
- Use RSA KX vulnerability for downgrade attack
  - Only requires server support for RSA KX
  - Works also on TLS 1.3 [JSS 15]
  - Require active MiTM attack
  - COOKIE?

ME WANT COOKIE! ALL COOKIES!

- Only 6% of connections use RSA KX
- Use RSA KX vulnerability for downgrade attack
  - Only requires server support for RSA KX
  - Works also on TLS 1.3 [JSS 15]
  - Require active MiTM attack
  - COOKIE?
- Time to finish attack < 30 sec

ME WANT COOKIE! ALL COOKIES!

- Only 6% of connections use RSA KX
- Use RSA KX vulnerability for downgrade attack
  - Only requires server support for RSA KX
  - Works also on TLS 1.3 [JSS 15]
  - Require active MiTM attack
  - COOKIE?
- Time to finish attack < 30 sec
  - Need many queries
  - Have time for < 600

Downgrade attack on Firefox

- We can prevent timeout in Firefox’s TLS handshakes using TLS warning alerts [ABDG+15]

Downgrade attack on Firefox

- We can prevent timeout in Firefox’s TLS handshakes using TLS warning alerts [ABDG+15]
- Do MiTM downgrade attack

Downgrade attack on Firefox

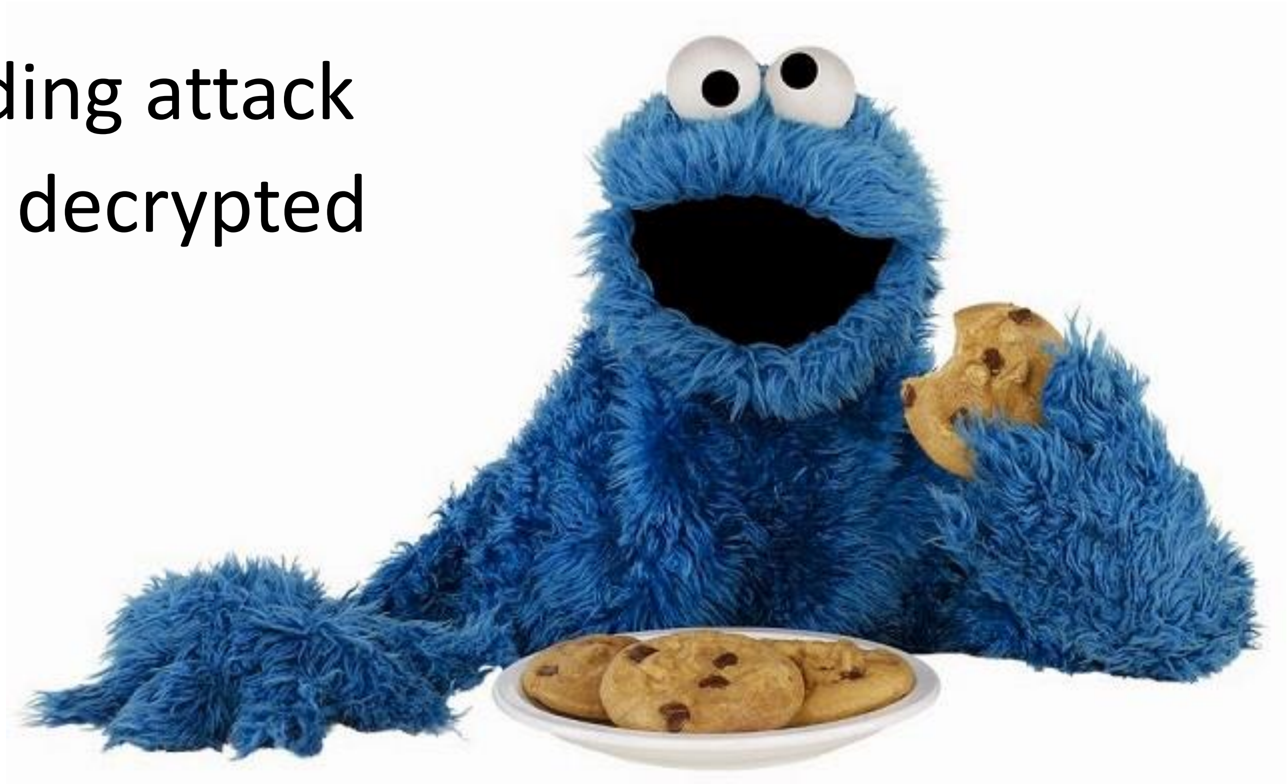
- We can prevent timeout in Firefox's TLS handshakes using TLS warning alerts [ABDG+15]
- Do MiTM downgrade attack
    - Keep session alive during padding attack

Downgrade attack on Firefox

- We can prevent timeout in Firefox’s TLS handshakes using TLS warning alerts [ABDG+15]
- Do MiTM downgrade attack
  - Keep session alive during padding attack
  - Finish the TLS handshake with decrypted premaster secret

Downgrade attack on Firefox

- We can prevent timeout in Firefox’s TLS handshakes using TLS warning alerts [ABDG+15]
- Do MiTM downgrade attack
  - Keep session alive during padding attack
  - Finish the TLS handshake with decrypted premaster secret
  - Cookie?



Downgrade attack on Firefox

- We can prevent timeout in Firefox's TLS handshakes using TLS warning alerts [ABDG+15]
- Do MiTM downgrade attack
  - Keep session alive during padding attack
  - Finish the TLS handshake with decrypted premaster secret
  - Cookie?
- The user will notice the delay

The Boost of the BEAST

- BEAST like attack can help!

The Boost of the BEAST

- BEAST like attack can help!
  - JavaScript in browser allows the attacker to repeatedly reopen connections in the background, without the user’s knowledge.

The Boost of the BEAST

- BEAST like attack can help!
  - JavaScript in browser allows the attacker to repeatedly reopen connections in the background, without the user’s knowledge.
  - At the start of each connection, the same session cookie is sent in the first packet

The Boost of the BEAST

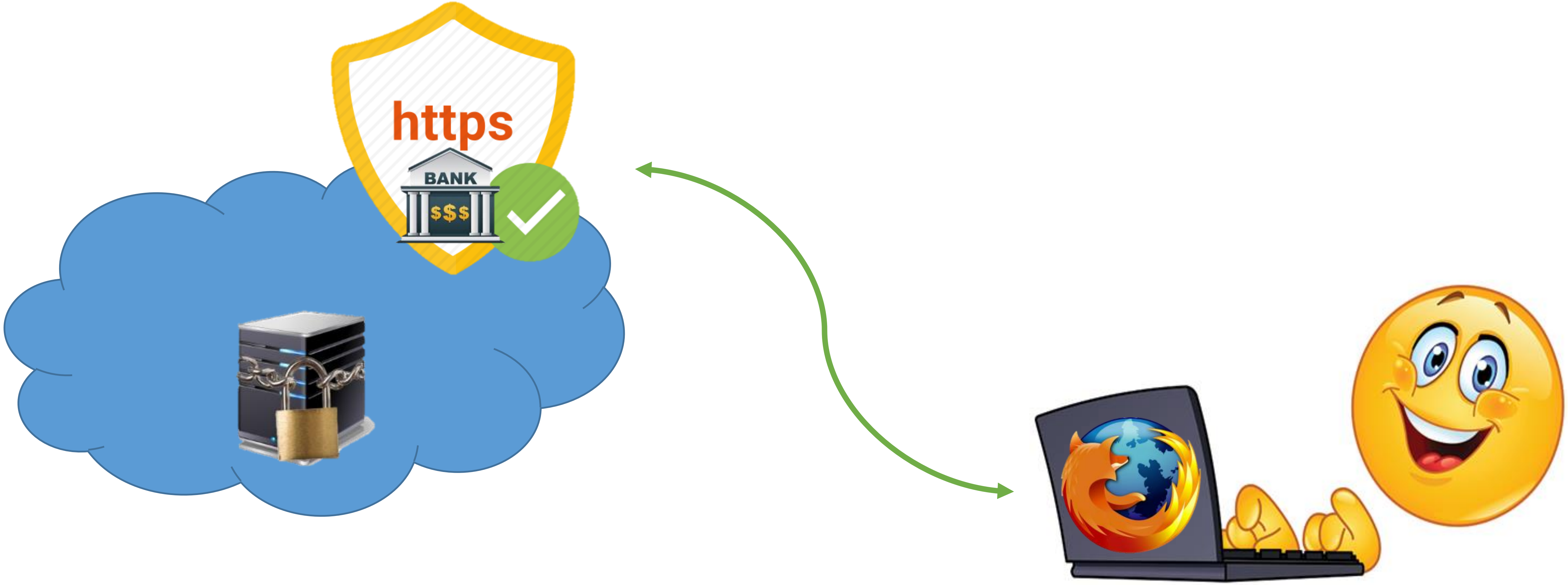
- BEAST like attack can help!
  - JavaScript in browser allows the attacker to repeatedly reopen connections in the background, without the user’s knowledge.
  - At the start of each connection, the same session cookie is sent in the first packet
  - Need to break just one connection

The Boost of the BEAST

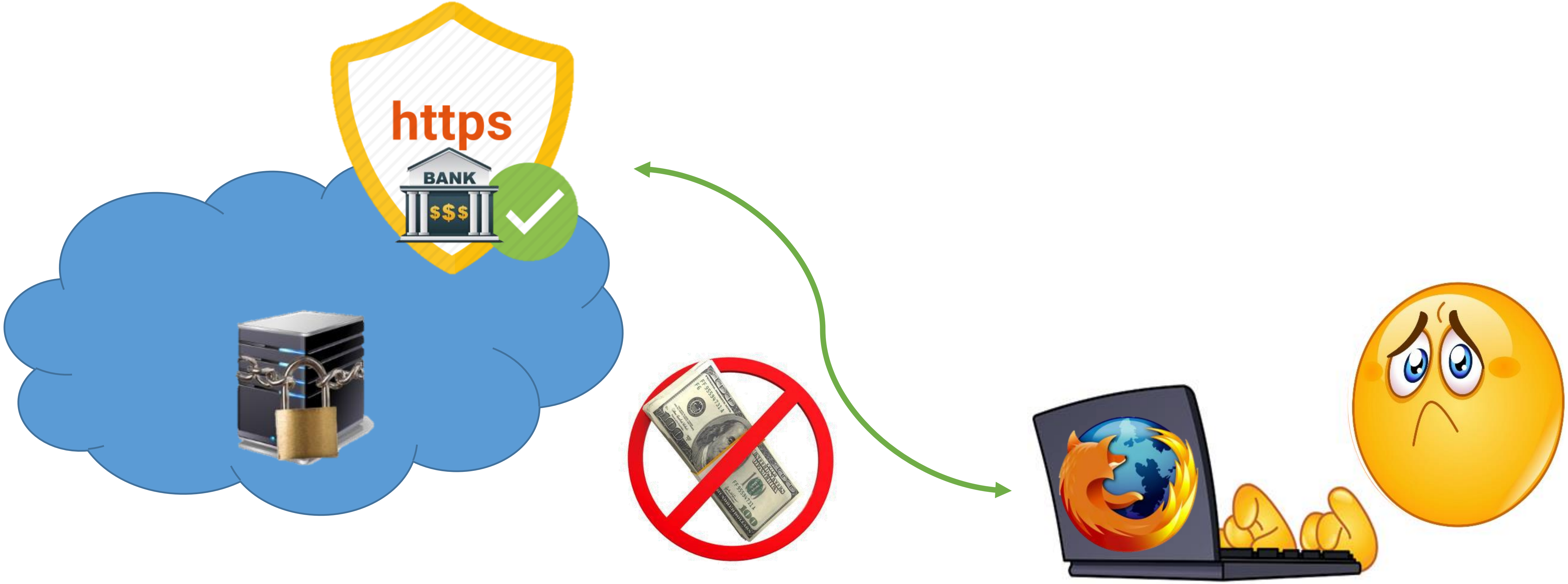
- BEAST like attack can help!
  - JavaScript in browser allows the attacker to repeatedly reopen connections in the background, without the user's knowledge.
  - At the start of each connection, the same session cookie is sent in the first packet
  - Need to break just one connection
  - COOKIE!

Attack Scenario Firefox:
MiTM + Cache timing side channel

Attack Scenario Firefox:
MiTM + Cache timing side channel



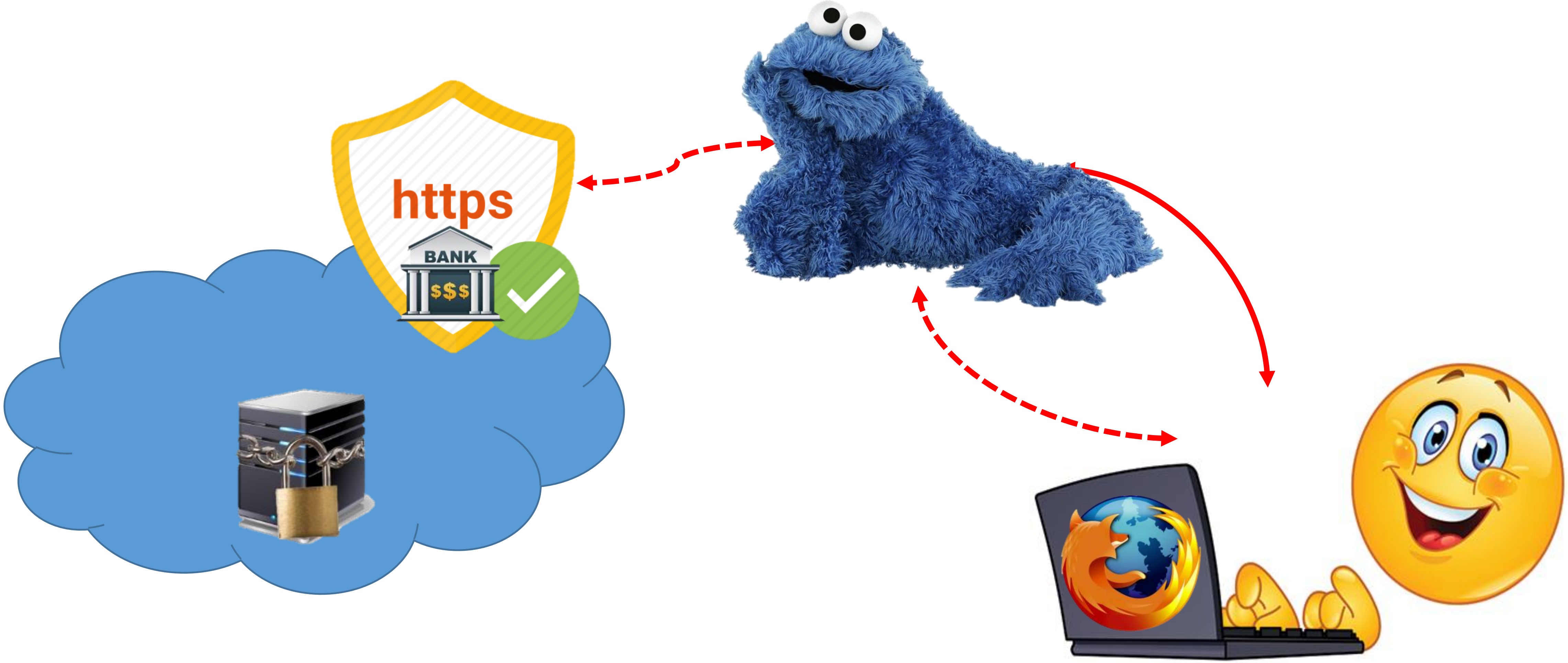
Attack Scenario Firefox:
MiTM + Cache timing side channel



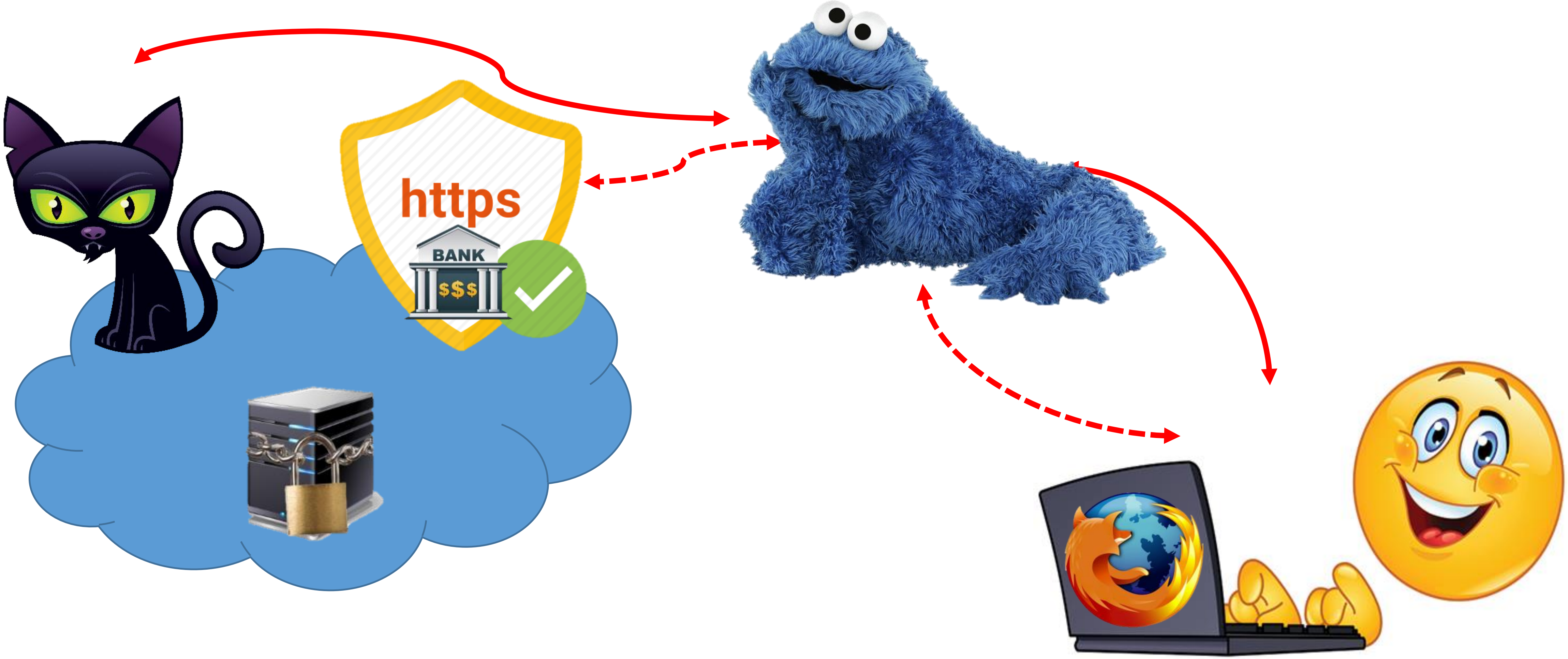
Attack Scenario Firefox: MiTM + Cache timing side channel

https
BANK
$$$
Win Big Prizes .COM

Attack Scenario Firefox:
MiTM + Cache timing side channel



Attack Scenario Firefox:
MiTM + Cache timing side channel



Attack Scenario Firefox: MiTM + Cache timing side channel

Attack Scenario Firefox:
MiTM + Cache timing side channel

Parallel Downgrade attack

- Most browsers timeout TLS handshake after 30 seconds

Parallel Downgrade attack

- Most browsers timeout TLS handshake after 30 seconds
- Many companies reuse certificate on multiple servers

Parallel Downgrade attack

- Most browsers timeout TLS handshake after 30 seconds
- Many companies reuse certificate on multiple servers
- We can parallelize the attack across multiple servers

Parallel Downgrade attack

- Most browsers timeout TLS handshake after 30 seconds
- Many companies reuse certificate on multiple servers
- We can parallelize the attack across multiple servers
    - Each server is a separate oracle

Parallel Downgrade attack

- Most browsers timeout TLS handshake after 30 seconds
- Many companies reuse certificate on multiple servers
- We can parallelize the attack across multiple servers
    - Each server is a separate oracle
    - Many previous works mention parallelization

Parallel Downgrade attack

- Most browsers timeout TLS handshake after 30 seconds
- Many companies reuse certificate on multiple servers
- We can parallelize the attack across multiple servers
  - Each server is a separate oracle
  - Many previous works mention parallelization
  - Cookie?

Parallel Downgrade attack

- Most browsers timeout TLS handshake after 30 seconds
- Many companies reuse certificate on multiple servers
- We can parallelize the attack across multiple servers
  - Each server is a separate oracle
  - Many previous works mention parallelization
  - Cookie?
- Need at least 2048 sequential adaptive queries
  - Have time for < 600

A little Manger background

- Assume we have the following Manger oracle

$$\mathsf{Ma}(c) = \begin{cases} 1 \text{ if } c^d \bmod N \text{ starts with } \mathsf{0x00} \\ 0 \text{ otherwise} \end{cases} .$$

A little Manger background

- Assume we have the following Manger oracle

$$\mathsf{Ma}(c) = \begin{cases} 1 \text{ if } c^d \bmod N \text{ starts with } \mathsf{0x00} \\ 0 \text{ otherwise} \end{cases} .$$

- We start with a blinding phase to find s such that

$$\mathsf{Ma}(c \cdot s^e \mod N) = \mathsf{Ma}((m \cdot s)^e \mod N) = 1$$

$$m \cdot s \mod N < 2^{\log_2 N - 8}$$

A little Manger background

- Assume we have the following Manger oracle

$$\mathsf{Ma}(c) = \begin{cases} 1 \text{ if } c^d \bmod N \text{ starts with } \mathsf{0x00} \\ 0 \text{ otherwise} \end{cases} .$$

- We start with a blinding phase to find s such that

$$\mathsf{Ma}(c \cdot s^e \mod N) = \mathsf{Ma}((m \cdot s)^e \mod N) = 1$$

$$m \cdot s \mod N < 2^{\log_2 N - 8}$$

0 N-1

A little Manger background

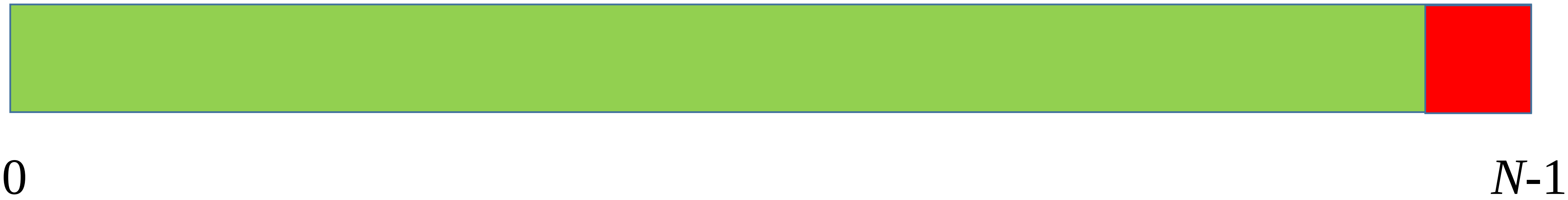
- Assume we have the following Manger oracle

$$\mathsf{Ma}(c) = \begin{cases} 1 \text{ if } c^d \bmod N \text{ starts with } \mathsf{0x00} \\ 0 \text{ otherwise} \end{cases} .$$

- We start with a blinding phase to find s such that

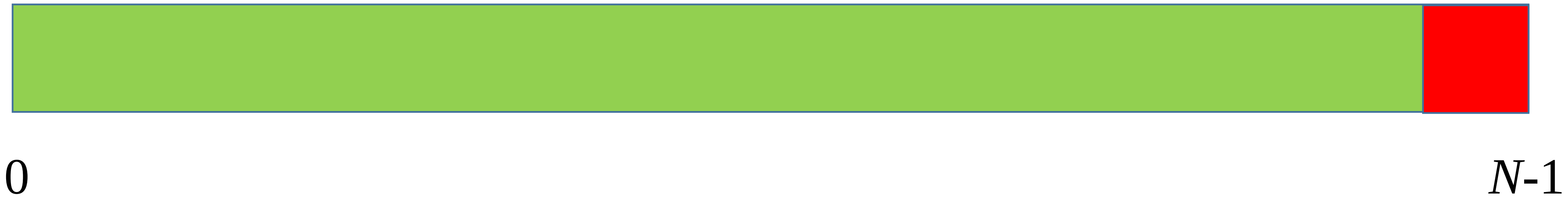
$$\mathsf{Ma}(c \cdot s^e \mod N) = \mathsf{Ma}((m \cdot s)^e \mod N) = 1$$

$$m \cdot s \mod N < 2^{\log_2 N - 8}$$



A little Manger background

- Iteratively reduce size of possible interval

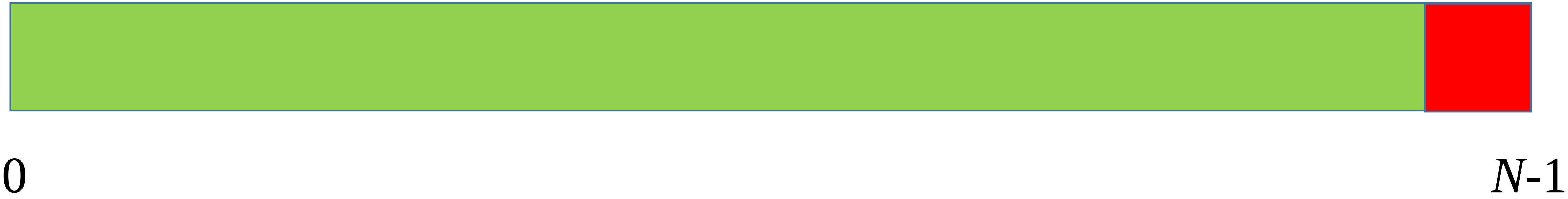


A little Manger background

- Iteratively reduce size of possible interval
- After additional i sequential queries we learn that

$$m \cdot s \mod N \in [a_i, b_i]$$

$$r_i = m \cdot s - a_i \mod N < 2^{\log_2 (b_i - a_i)}$$

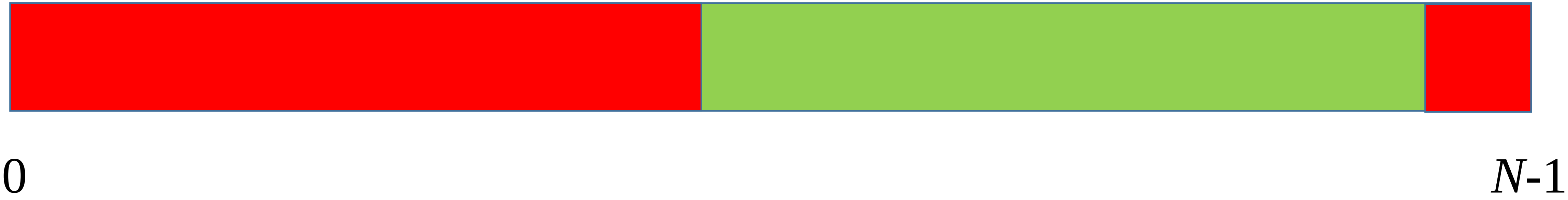


A little Manger background

- Iteratively reduce size of possible interval
- After additional i sequential queries we learn that

$$m \cdot s \mod N \in [a_i, b_i]$$

$$r_i = m \cdot s - a_i \mod N < 2^{\log_2 (b_i - a_i)}$$

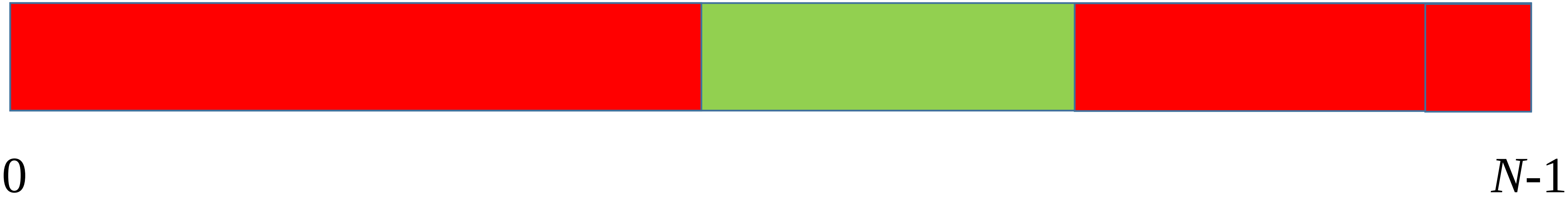


A little Manger background

- Iteratively reduce size of possible interval
- After additional i sequential queries we learn that

$$m \cdot s \mod N \in [a_i, b_i]$$

$$r_i = m \cdot s - a_i \mod N < 2^{\log_2 (b_i - a_i)}$$

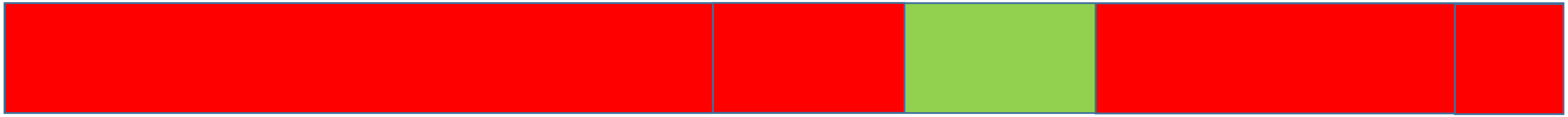


A little Manger background

- Iteratively reduce size of possible interval
- After additional i sequential queries we learn that

$$m \cdot s \mod N \in [a_i, b_i]$$

$$r_i = m \cdot s - a_i \mod N < 2^{\log_2 (b_i - a_i)}$$

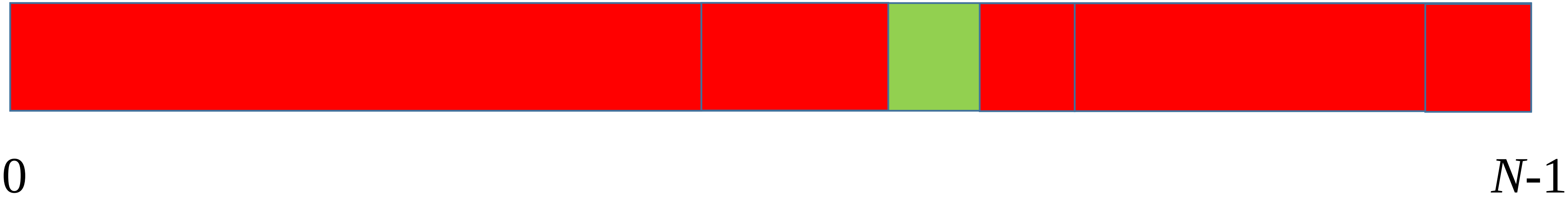


A little Manger background

- Iteratively reduce size of possible interval
- After additional i sequential queries we learn that

$$m \cdot s \mod N \in [a_i, b_i]$$

$$r_i = m \cdot s - a_i \mod N < 2^{\log_2 (b_i - a_i)}$$

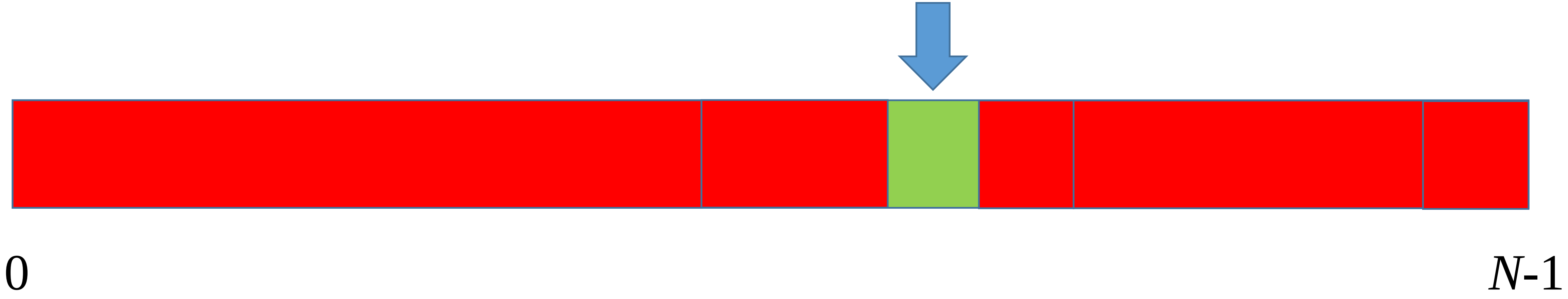


A little Manger background

- Iteratively reduce size of possible interval
- After additional i sequential queries we learn that

$$m \cdot s \mod N \in [a_i, b_i]$$

$$r_i = m \cdot s - a_i \mod N < 2^{\log_2 (b_i - a_i)}$$



The Cookie Lattice

- Run k attacks in parallel with i sequential queries each

$$r_{j,i} = m \cdot s_j - a_{j,i} \mod N < 2^{\log_2 (b_{j,i} - a_{j,i})}$$

The Cookie Lattice

- Run k attacks in parallel with i sequential queries each

$$r_{j,i} = m \cdot s_j - a_{j,i} \mod N < 2^{\log_2(b_{j,i} - a_{j,i})}$$

- Similar to Boneh & Venkatesan’s Hidden Number Problem

The Cookie Lattice

- Run k attacks in parallel with i sequential queries each

$$r_{j,i} = m \cdot s_j - a_{j,i} \mod N < 2^{\log_2 (b_{j,i}-a_{j,i})}$$

- Similar to Boneh & Venkatesan's Hidden Number Problem
- Finding m is reduced to CVP that we can embed in a SVP lattice and solve with LLL

$$M^i = \begin{bmatrix} s_1 & s_2 & s_3 & \dots & s_k & 0 \\ N & 0 & 0 & \dots & 0 & 0 \\ 0 & N & 0 & \dots & 0 & 0 \\ 0 & 0 & N & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & N & 0 \\ a_{1,i} & a_{2,i} & a_{3,i} & \dots & a_{k,i} & N \cdot (k-1)/k \end{bmatrix}$$

The Cookie Lattice

- Run k attacks in parallel with i sequential queries each

$$r_{j,i} = m \cdot s_j - a_{j,i} \mod N < 2^{\log_2 (b_{j,i} - a_{j,i})}$$

- Similar to Boneh & Venkatesan's Hidden Number Problem
- Finding m is reduced to CVP that we can embed in a SVP lattice and solve with LLL

- We need just 5 servers to decrypt 2048 bit RSA using a Manger oracle

$$M^i = \begin{bmatrix} s_1 & s_2 & s_3 & \dots & s_k & 0 \\ N & 0 & 0 & \dots & 0 & 0 \\ 0 & N & 0 & \dots & 0 & 0 \\ 0 & 0 & N & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & N & 0 \\ a_{1,i} & a_{2,i} & a_{3,i} & \dots & a_{k,i} & N \cdot (k-1)/k \end{bmatrix}$$

The Cookie Lattice

- Run k attacks in parallel with i sequential queries each

$$r_{j,i} = m \cdot s_j - a_{j,i} \mod N < 2^{\log_2 (b_{j,i} - a_{j,i})}$$

- Similar to Boneh & Venkatesan's Hidden Number Problem
- Finding m is reduced to CVP that we can emb lattice and solve with LLL

- We need just 5 servers to decrypt 2048 bit RSA using a Manger oracle

The Cookie Lattice Tradeoff

- The initial blinding phase is more “expensive” per bit

The Cookie Lattice Tradeoff

- The initial blinding phase is more “expensive” per bit
- The parallel attack requires more queries!

The Cookie Lattice Tradeoff

- The initial blinding phase is more “expensive” per bit
- The parallel attack requires more queries!
- So why do we do it?

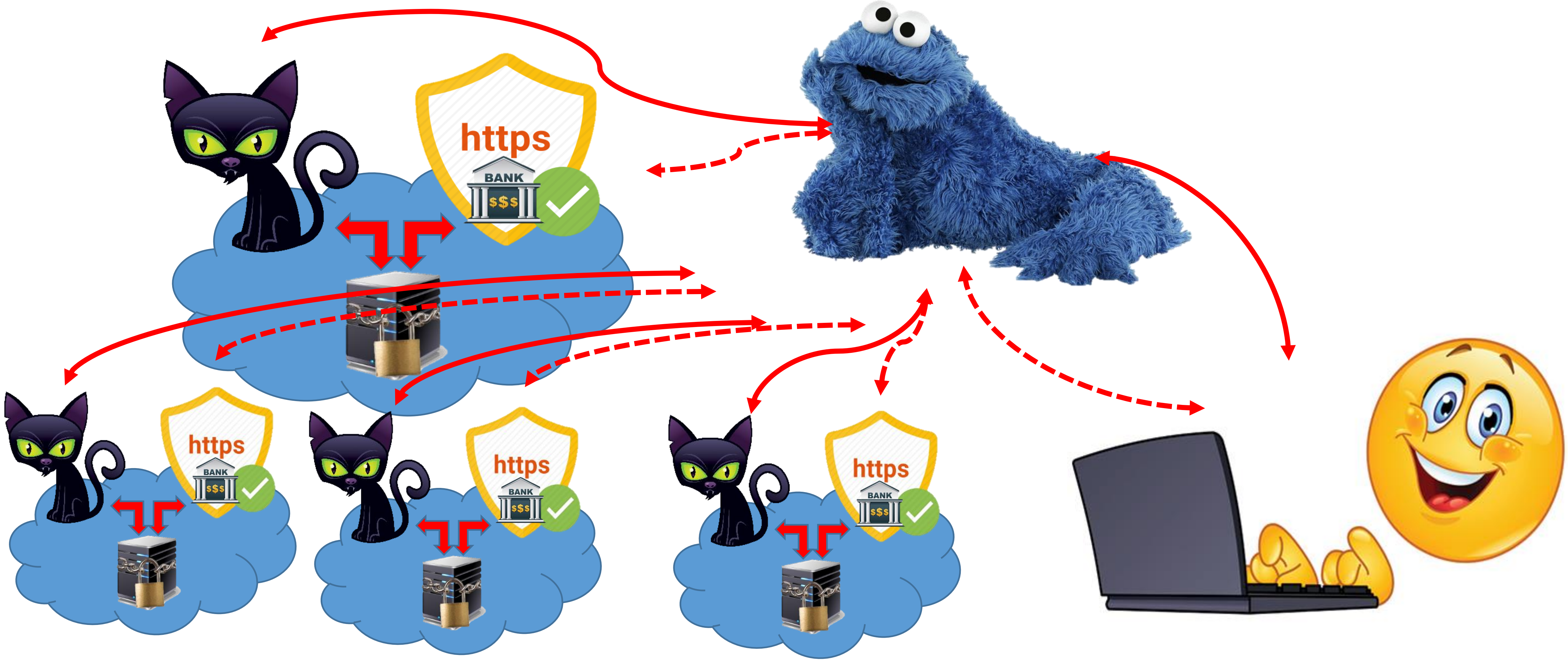
The Cookie Lattice Tradeoff

- The initial blinding phase is more “expensive” per bit
- The parallel attack requires more queries!
- So why do we do it?
  - Tradeoff between the total number of queries and number of sequential queries

The Cookie Lattice Tradeoff

- The initial blinding phase is more “expensive” per bit
- The parallel attack requires more queries!
- So why do we do it?
  - Tradeoff between the total number of queries and number of sequential queries
  - Allows us to finish attack in less than 30 seconds

Attack Scenario Parallel: MiTM + Cache timing side channel



Attack Scenario Parallel: MiTM + Cache timing side channel

$$M^i = \begin{bmatrix} s_1 & s_2 & s_3 & \dots & s_k & 0 \\ N & 0 & 0 & \dots & 0 & 0 \\ 0 & N & 0 & \dots & 0 & 0 \\ 0 & 0 & N & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & N & 0 \\ a_1^i & a_2^i & a_3^i & \dots & a_k^i & N \cdot (k-1)/k \end{bmatrix}$$

Attack Scenario Parallel: MiTM + Cache timing side channel

Our results

- New Techniques for Microarchitectural Padding Oracle Attacks, vulnerabilities in 7 out 9 implementations
  - PoC for Manger and Bleichenbacher attacks

Our results

- New Techniques for Microarchitectural Padding Oracle Attacks, vulnerabilities in 7 out 9 implementations
  - PoC for Manger and Bleichenbacher attacks

- Parallelization for downgrade attack
  - PoC for Manger parallelization using LLL

99% OF THE WORLDS COOKIES ARE CONSUMED BY 1% OF THE MONSTERS
LIFE
# OCCUPY SESAME STREET

Disclosure

- We disclosed to:
  - OpenSSL, Mozilla’s NSS, Amazon’s s2n, Apple’s CoreTLS, mbed TLS, wolfSSL, GnuTLS
- All have patched their code, with various levels of success
- Lots of stories…

Recommendation

- Many recommendations for several layers of mitigations in the paper
  - Bottom line **Don’t use RSA KX**
  - It has failed us too many times

Recommendation

- Many recommendations for several layers of mitigations in the paper
  - Bottom line **Don’t use RSA KX**
  - It has failed us too many times
- If you really really really must
  - Separate your certificates!
  - …

Questions?

- Paper website
  https://cat.eyalro.net

- Any questions?