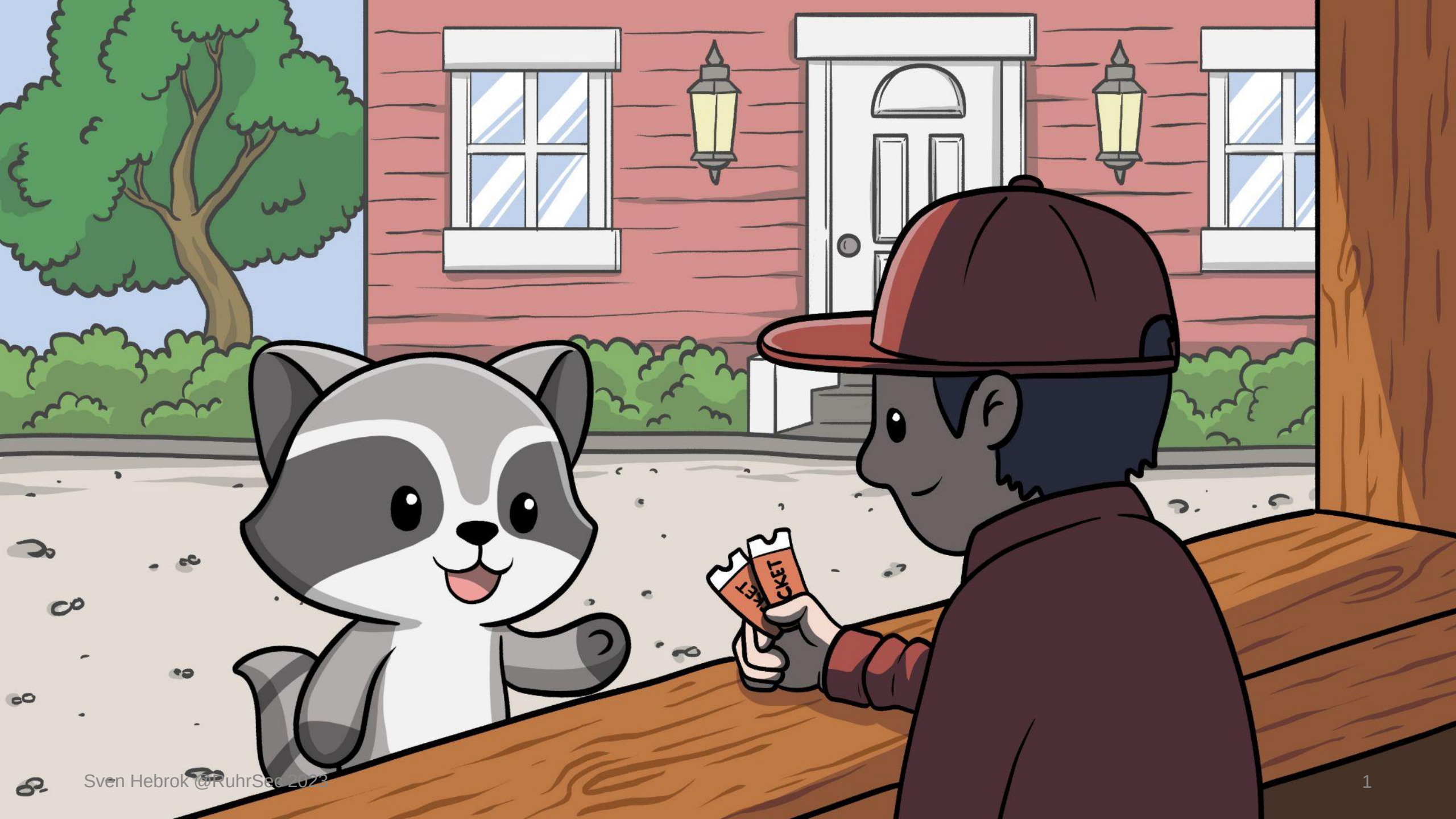

CKET

We Really Need to Talk About Session Tickets

We Need to Talk About Session Tickets

https://words.filippo.io/we-need-to-talk-about-session-tickets/

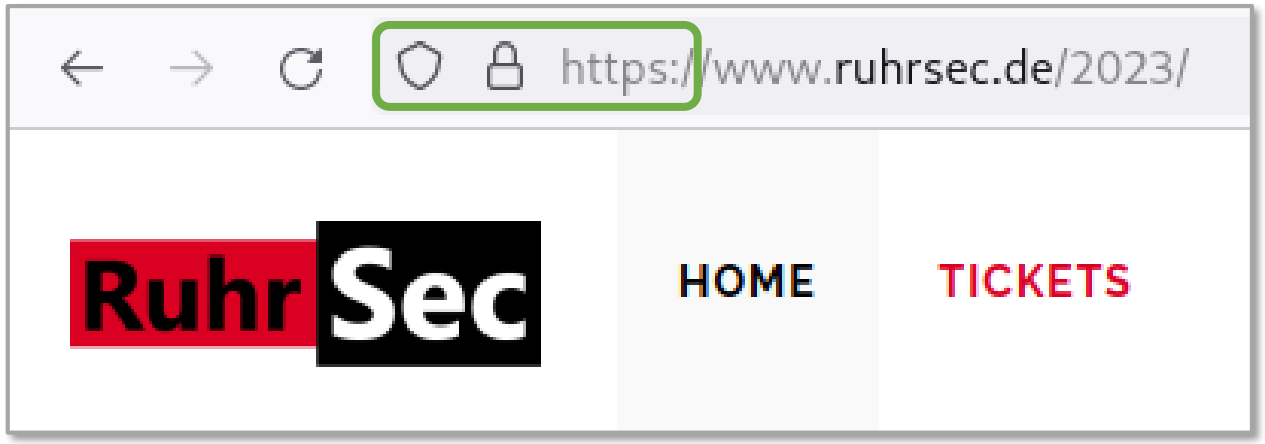
We *Really* Need to Talk About Session Tickets

Sven Hebrok, Simon Nachtigall, Marcel Maehren, Nurullah Erinola,

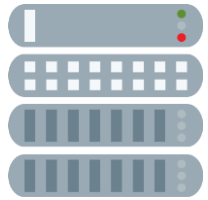
Robert Merget, Juraj Somorovsky, and Jörg Schwenk

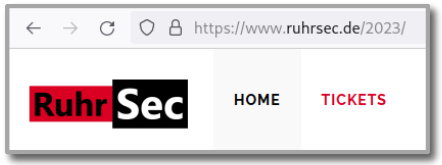
We *Really* Need to Talk About Session Tickets

We *Really* Need to Talk About TLS Session Tickets



We *Really* Need to Talk About TLS Session Tickets





Master Secret

Handshake (incl. Key Ex)

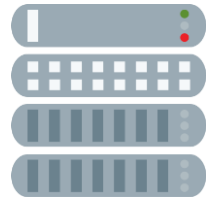
Session Keys

Application

Handshake (incl. Key Ex)

Application

TLS Handshake



Hello

Hello

Certificate

Key Exchange

Hello Done

Key Exchange

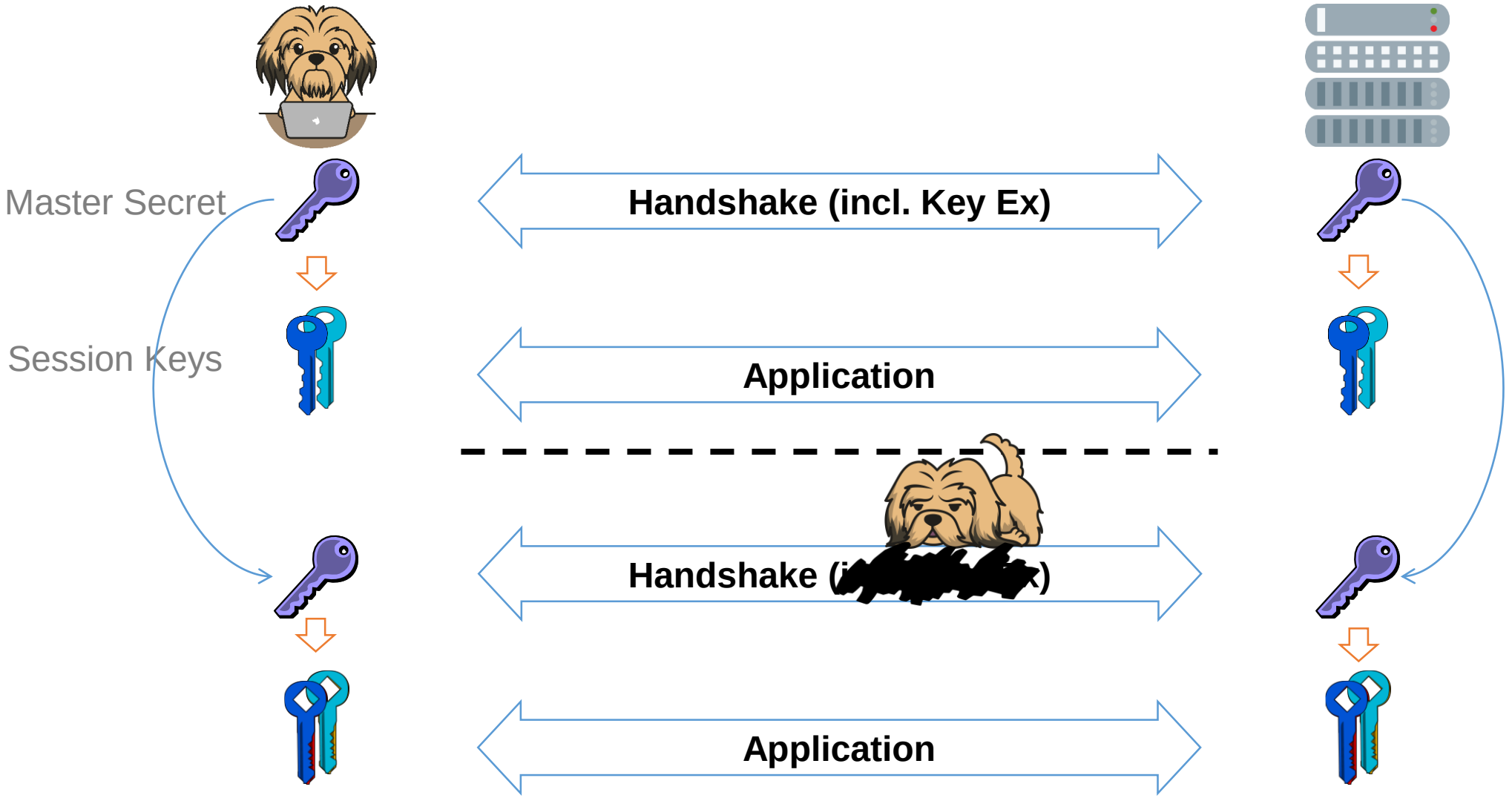
...

Application

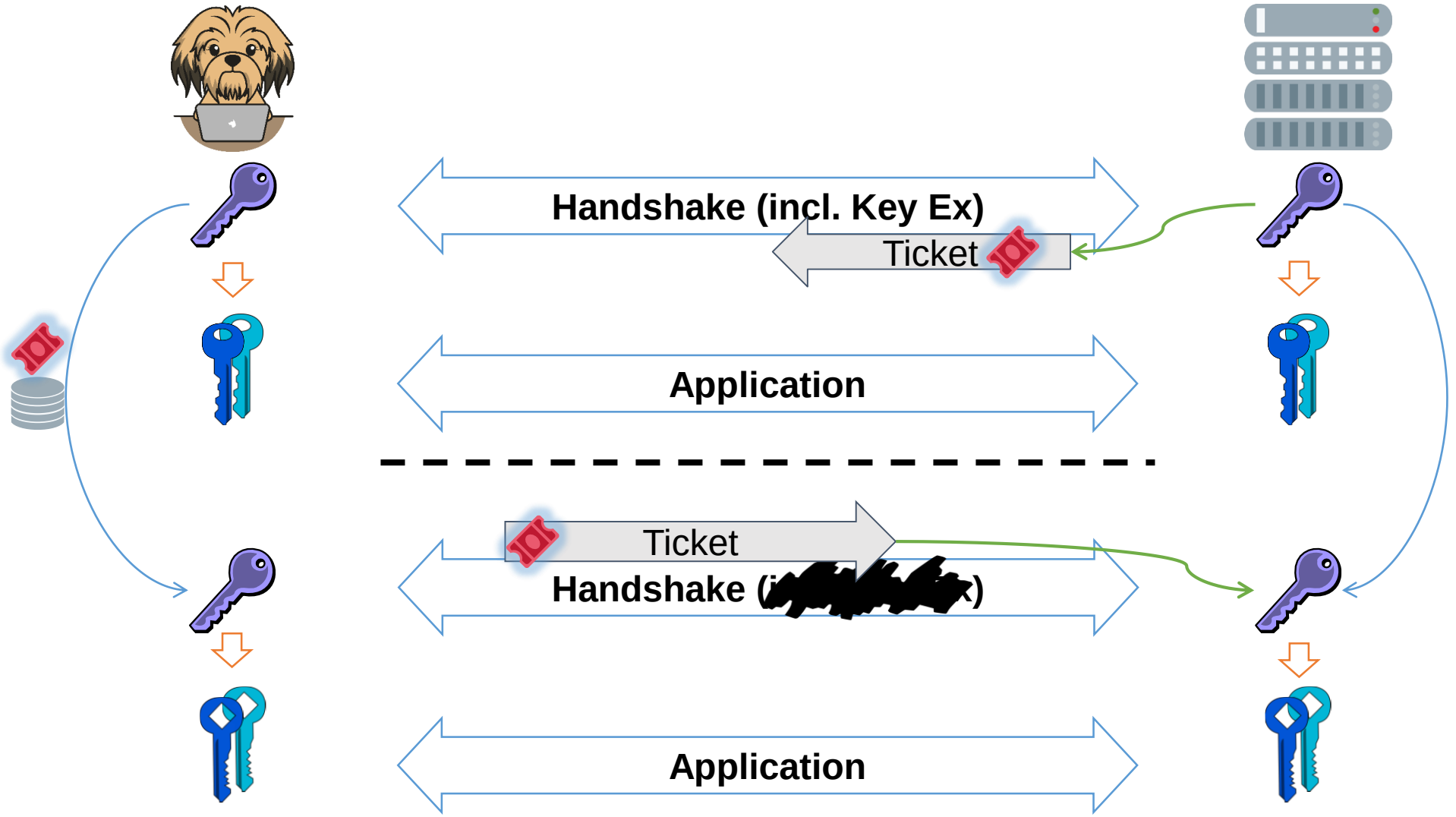
(EC)DHE
(EC)DH
RSA

RSA/DSS
Signature

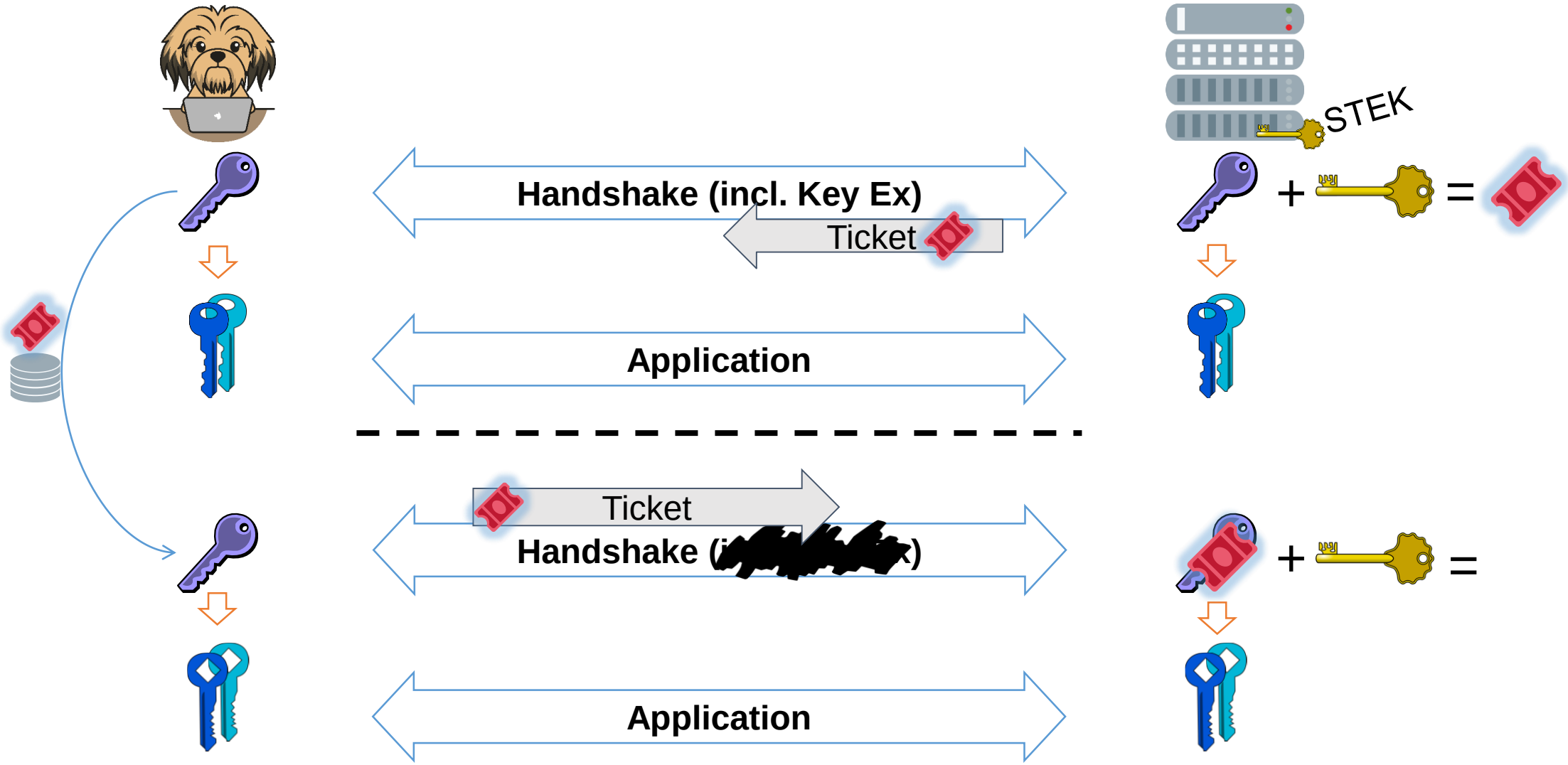
We *Really* Need to Talk About TLS Session Tickets



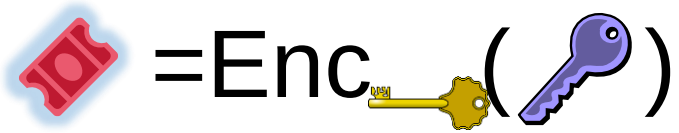
We *Really* Need to Talk About TLS Session Tickets

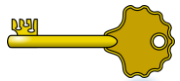


We *Really* Need to Talk About TLS Session Tickets

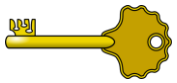


We *Really* Need to Talk About TLS Session Tickets



Server stores one  - used for all clients

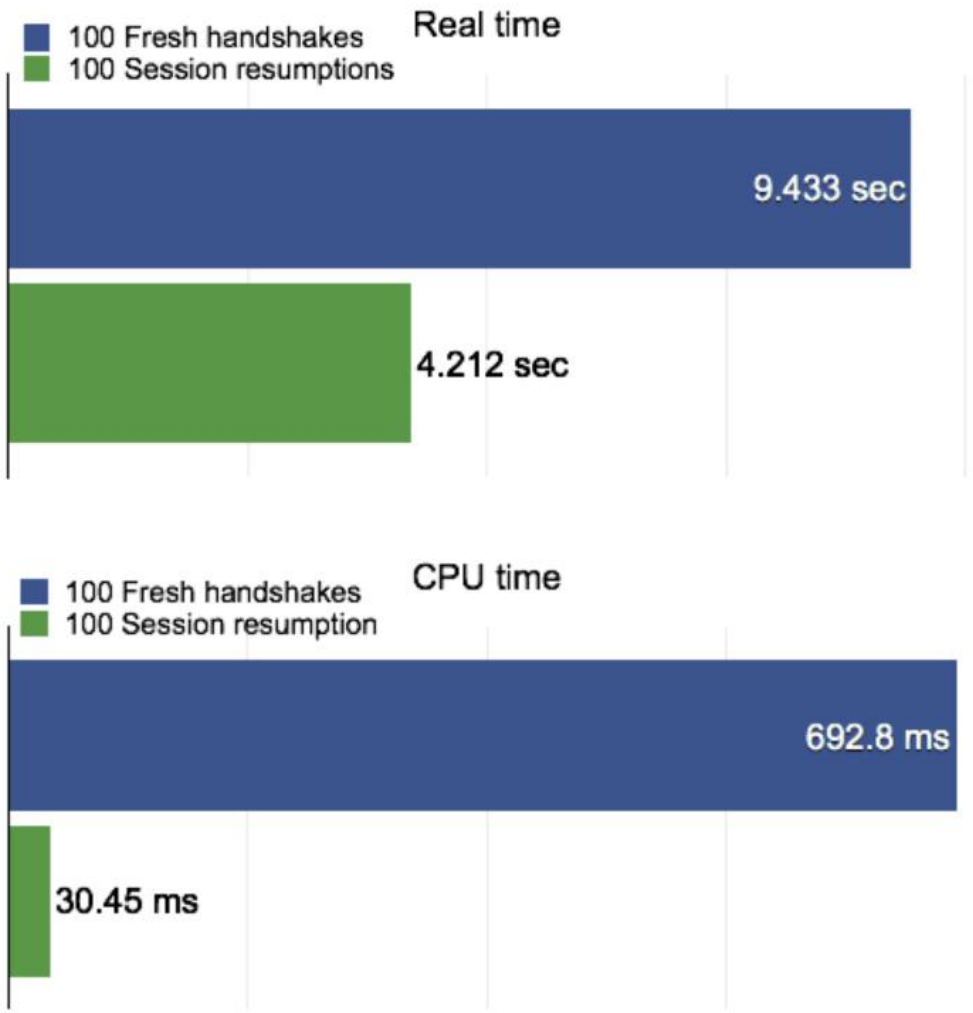
Client stores  and 

 STEK (**S**ession **T**icket **E**ncryption **K**ey)

We *Really* Need to Talk About TLS Session Tickets

In Practice

- near 100% Browser Support
- ~75% Server Support
- −50% connection establishment time
- −95% CPU time



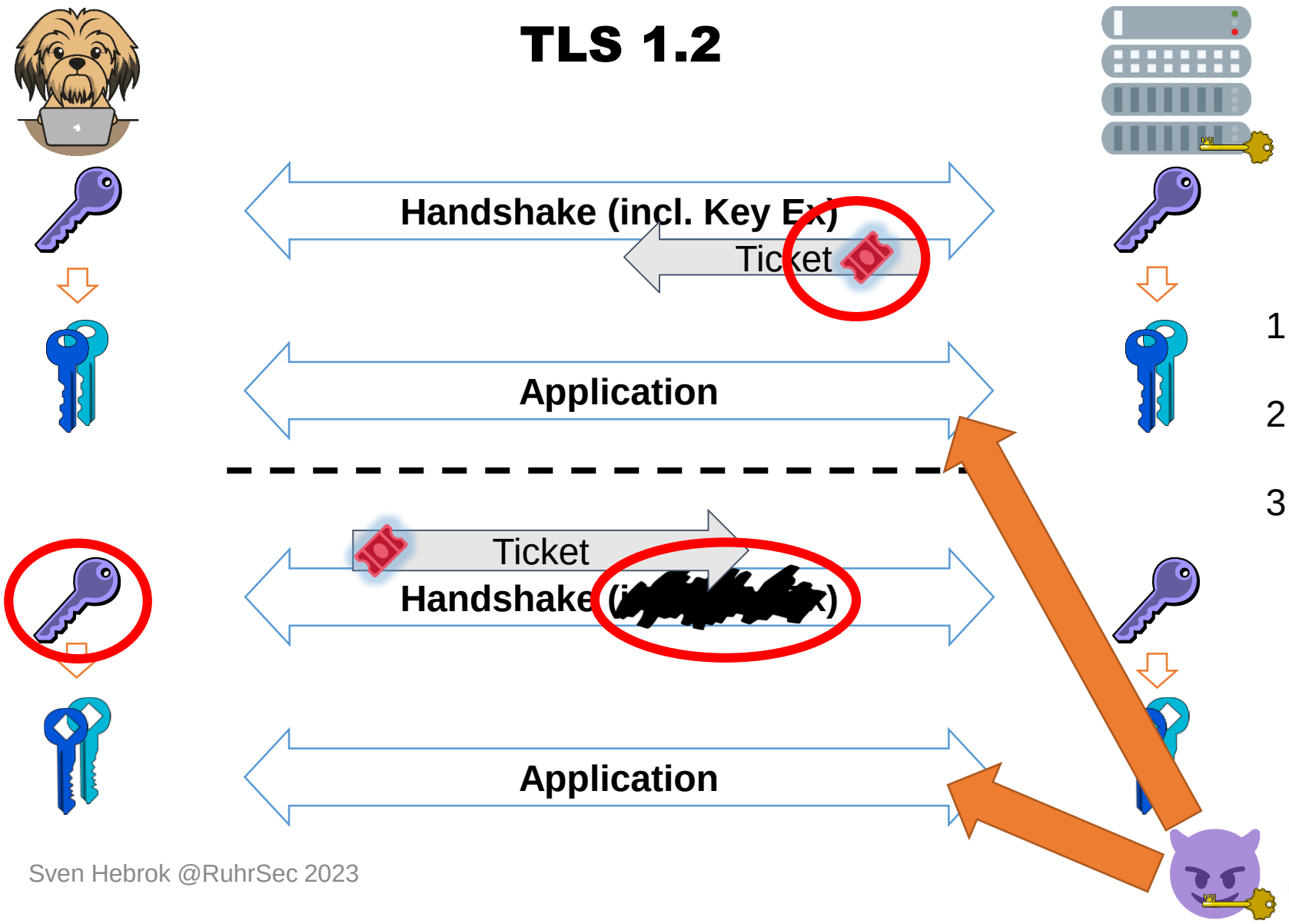
https://blog.cloudflare.com/tls-session-resumption-full-speed-and-secure/

We Need to Talk About Session Tickets

https://words.filippo.io/we-need-to-talk-about-session-tickets/

We Need to Talk About Session Tickets

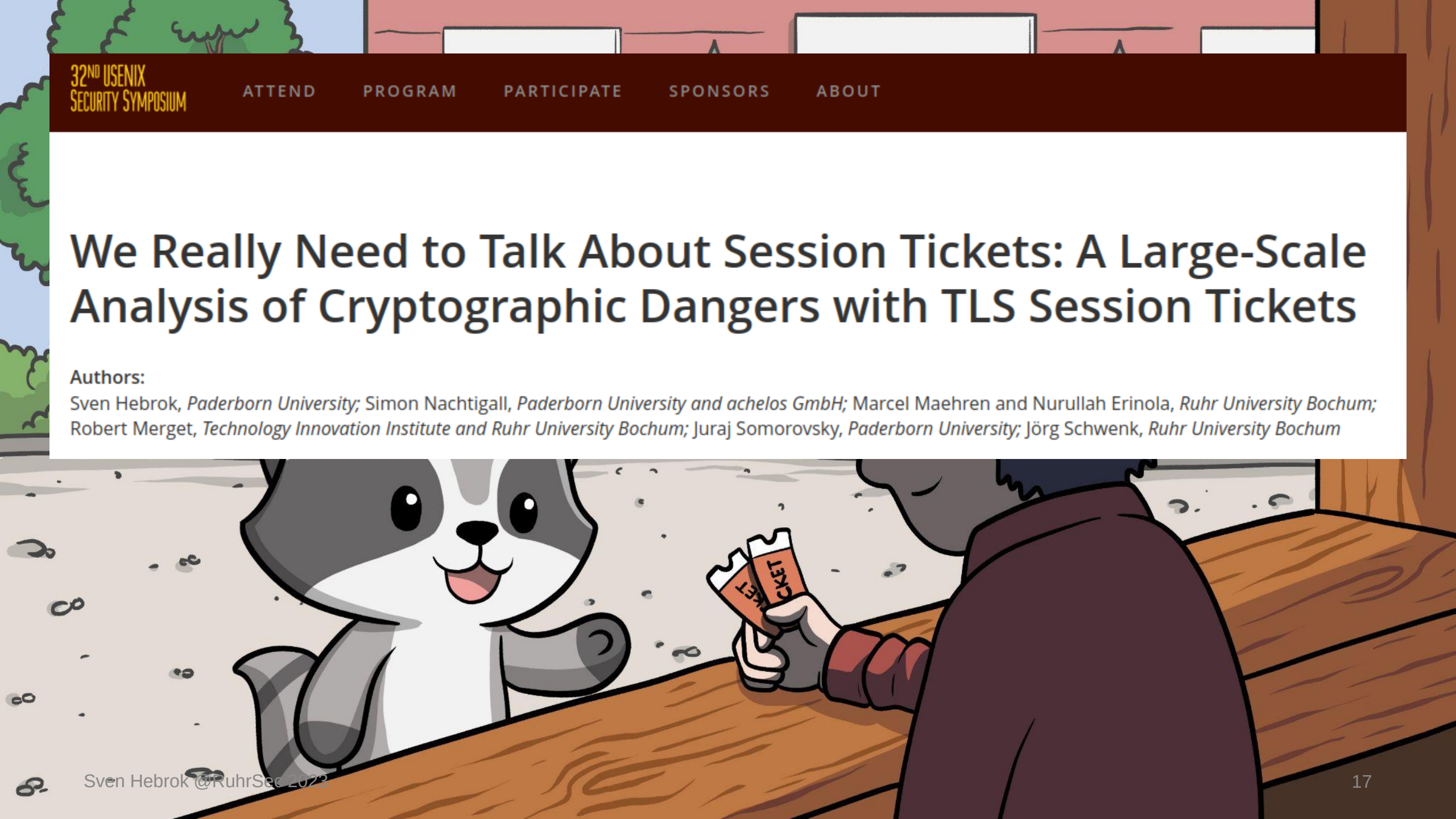
TLS 1.2



1. Can decrypt recorded sessions
2. Same secret reused
3. Tickets sent in plaintext

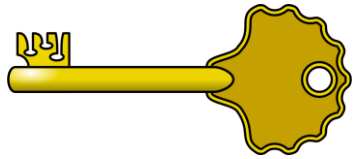
TLS Session Tickets
Brief Summary

- Speed up handshake
  - No key exchange
  - No certificate
- Encrypted using STEK
  - Known only to server
- STEK compromise catastrophic
  - Passive traffic decryption
  - Decrypt previous and future sessions
  - Active server impersonation


32ND USENIX SECURITY SYMPOSIUM
ATTEND
PROGRAM
PARTICIPATE
SPONSORS
ABOUT
We Really Need to Talk About Session Tickets: A Large-Scale Analysis of Cryptographic Dangers with TLS Session Tickets
Authors:
Sven Hebrok, Paderborn University; Simon Nachtigall, Paderborn University and achelos GmbH; Marcel Maehren and Nurullah Erinola, Ruhr University Bochum; Robert Merget, Technology Innovation Institute and Ruhr University Bochum; Juraj Somorovsky, Paderborn University; Jörg Schwenk, Ruhr University Bochum
TICKET
TICKET

Motivation: GnuTLS

CVE-ID	
CVE-2020-13777	Learn more at National Vulnerability Database (NVD) • CVSS Severity Rating • Fix Information • Vulnerable Software Versions • SCAP Mappings • CPE Information
Description	
GnuTLS 3.6.x before 3.6.14 uses incorrect cryptography for encrypting a session ticket (a loss of confidentiality in TLS 1.2, and an authentication bypass in TLS 1.3). The earliest affected version is 3.6.4 (2018-09-24) because of an error in a 2018-09-18 commit. Until the first key rotation, the TLS server always uses wrong data in place of an encryption key derived from an application.	

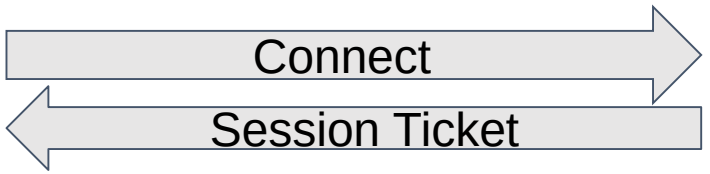


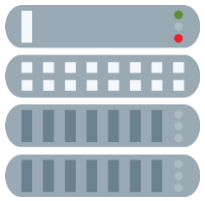
= 0x00000000000000000000000000000000

How widespread is something like this?

➔ Scan servers in the wild

Ticket Format?





How to determine whether STEK=000000?

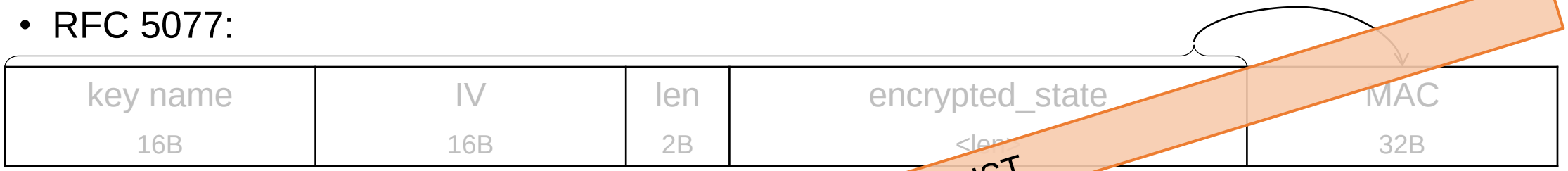
➔ decrypt with key=000000

What to decrypt?
Where's the IV?
Where's the Ciphertext?

How do tickets actually work?

Session Tickets on the Byte Level

- RFC 5077:



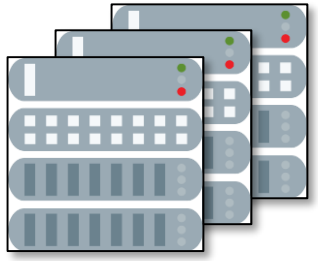
- AES-128-CBC
- HMAC-SHA-256
- Encrypt-then-Mac
- Regular key rotation

only RECOMMENDED, no MUST

Implementations could do anything

Our Plan

```
uint8_t *ptr;
if (!CBB_add_bytes(out, key_name, 16) ||
    !CBB_add_bytes(out, iv, EVP_CIPHER_CTX_iv_length(ctx.get())) ||
    !CBB_reserve(out, &ptr, session_len + EVP_MAX_BLOCK_LENGTH)) {
  return 0;
}
```



1. Analyze open-source implementations

- Ticket format
- Algorithms
- Look for immediate issues

2. Large-scale analysis

Open Source Analysis Algorithms

RFC 5077	AES-128-CBC	HMAC-SHA-256
BoringSSL	**AES-128-CBC**	**HMAC-SHA-256**

Open Source Analysis Algorithms

RFC 5077	AES-128-CBC	HMAC-SHA-256
BoringSSL	**AES-128-CBC**	**HMAC-SHA-256**
Botan	**AES**-256-GCM	(GMAC)
GnuTLS	**AES**-256-**CBC**	**HMAC-SHA**-1
GoTLS	**AES-128**-CTR	**HMAC-SHA-256**
MatrixSSL (TLS 1.2)	**AES**-256-**CBC**	**HMAC-SHA-256**
MatrixSSL (TLS 1.3)	**AES**-256-GCM	(GMAC)
mbedTLS	**AES-128**/256-GCM/CCM	(GMAC/CBCMAC)
OpenSSL	**AES**-256-**CBC**	**HMAC-SHA-256**
Rustls	ChaCha20	Poly1305
s2n	**AES**-256-GCM	(GMAC)
Apache	**AES-128-CBC**	**HMAC-SHA-256**
Nginx	**AES-128**/256-**CBC**	**HMAC-SHA-256**
OpenLiteSpeed	**AES-128-CBC**	**HMAC-SHA-256**

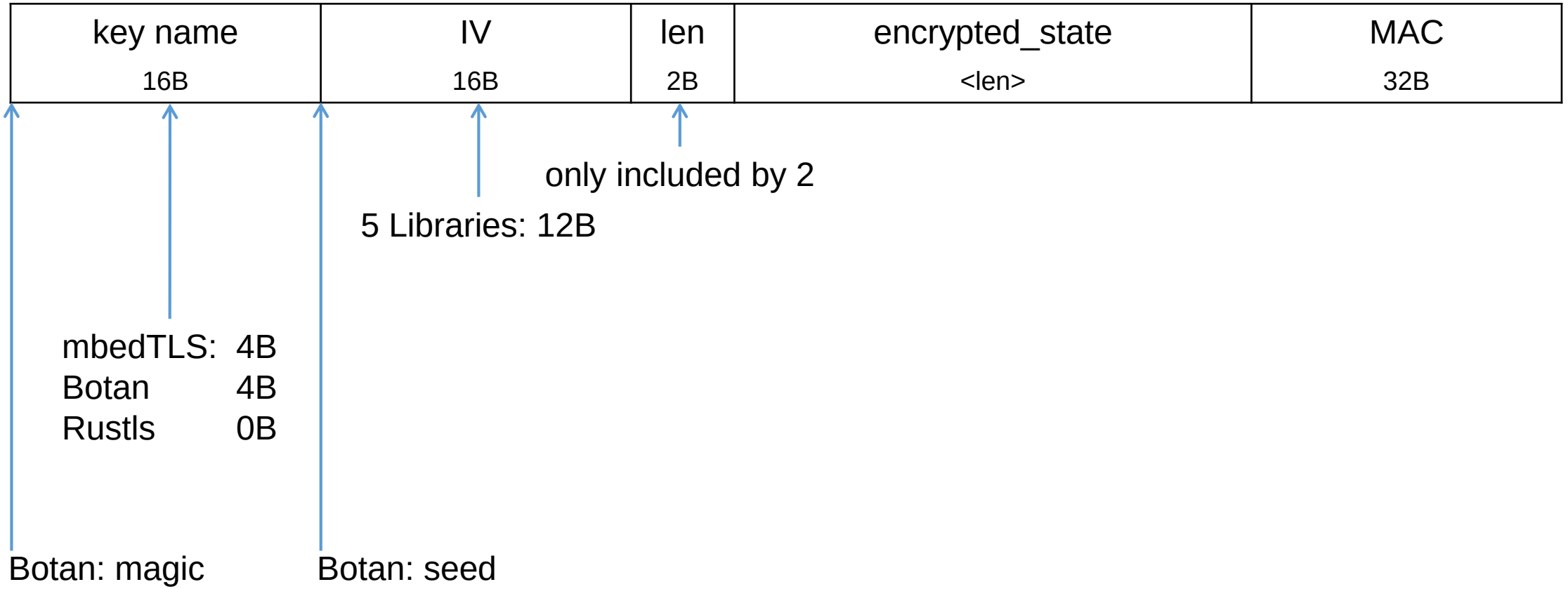
As Recommended

Complexity

Key Wearout Possible - Addressed

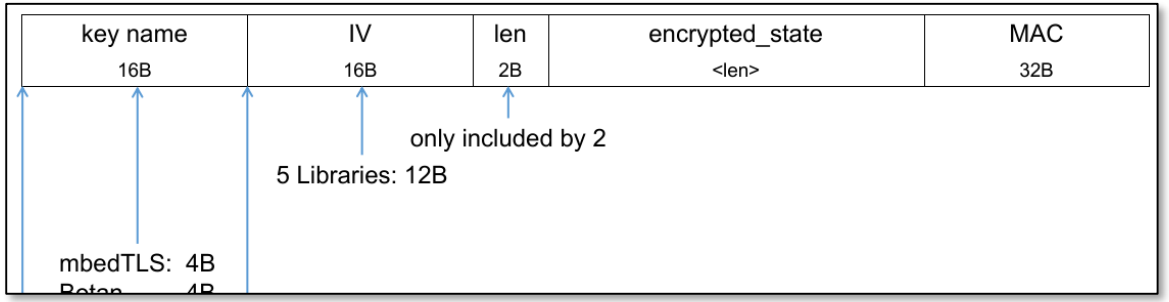
Weaker than Recommended

Open Source Analysis
Format



Open-Source Analysis Conclusion

RFC 5077	AES-128-CBC	HMAC-SHA-256
BoringSSL	**AES-128-CBC**	**HMAC-SHA-256**
Botan	**AES**-256-GCM	(GMAC)
GnuTLS	**AES**-256-**CBC**	**HMAC-SHA**-1
GoTLS	**AES-128**-CTR	**HMAC-SHA-256**
MatrixSSL (TLS 1.2)	**AES**-256-**CBC**	**HMAC-SHA-256**



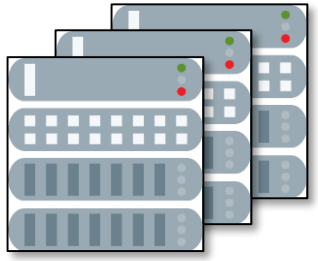
- Some complexity
- One weaker algorithm

- Close to recommended format
  - Some fields shorter
  - IV/Nonce before ciphertext
  - Encrypt-then-MAC

no immediate issues

Our Plan

```
uint8_t *ptr;
if (!CBB_add_bytes(out, key_name, 16) ||
    !CBB_add_bytes(out, iv, EVP_CIPHER_CTX_iv_length(ctx.get())) ||
    !CBB_reserve(out, &ptr, session_len + EVP_MAX_BLOCK_LENGTH)) {
  return 0;
}
```



1. Analyze open-source implementations
- Ticket format
- Algorithms
- Look for immediate issues

2. Large-scale analysis
- Propose potential pitfalls
- How to gather tickets
- How to analyze tickets
- Perform Scan

Large Scale Evaluation
Potential Pitfalls

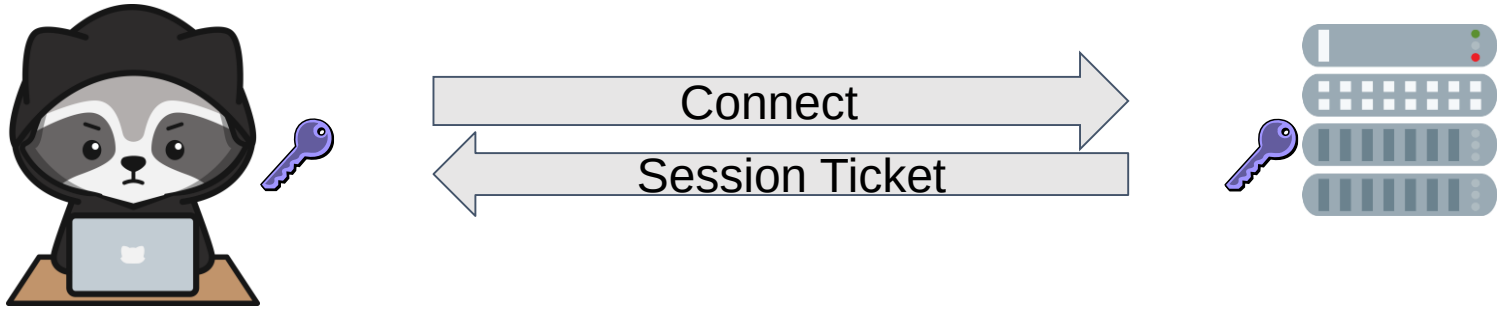
Offline
just need ticket

- Plaintext session tickets
  - Would still work
- Weak STEK (enc or auth)
  - 0x000000, 0x010203, ...
  - Also check weak algorithms
- Reused Keystream
  - e.g. CTR/GCM with reused nonce

Online
observe sever behavior

- Authentication Issues
  - Bitflips
  - CBC Padding Oracle

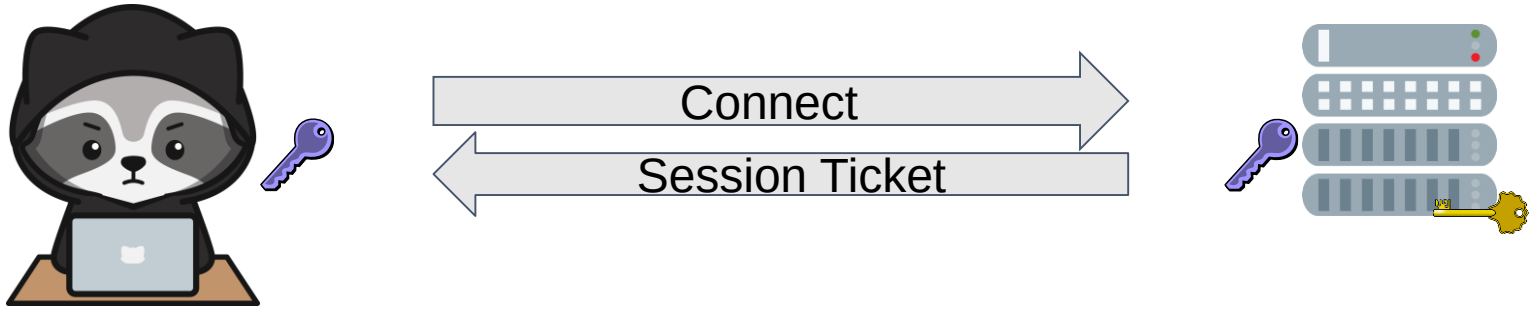
Large Scale Evaluation
Plaintext Ticket




```
▾ Handshake Protocol: New Session Ticket
    Handshake Type: New Session Ticket (4)
    Length: 235
  ▾ TLS Session Ticket
      Session Ticket Lifetime Hint: 100800 seconds (1 day, 4 hours)
      Session Ticket Length: 229
      Session Ticket: 02d32b3f673ba516106107535a8f78ad158f3134dca563e7…
▸ TLSv1.2 Record Layer: Change Cipher Spec Protocol: Change Cipher Spec
▾ TLSv1.2 Record Layer: Handshake Protocol: Encrypted Handshake Message
    Content Type: Handshake (22)
    Version: TLS 1.2 (0x0303)
```

```
0050  c0 00 e5 02 d3 2b 3f 67  3b a5 16 10 61 07 53 5a   ···· ··+?g ;···a·SZ
0060  8f 78 ad 15 8f 31 34 dc  a5 63 e7 70 31 27 6d 73   ·x···14· ·c·p1'ms
0070  13 1c 6a 0c 09 bc b9 71  41 fd 60 fc 1a 4b dd 7d   ··j····q A·`··K·}
0080  6d ac 0d 59 cb 45 48 c6  3e b0 02 3e 4f e7 f7 45   m··Y·EH· >··>O··E
0090  73 28 eb 73 93 58 7a c0  68 f7 c6 d4 d0 20 66 a7   s(·s·Xz· h···· f·
00a0  77 8d 97 2e 0f f4 e6 2a  d7 a3 59 db b2 a2 36 61   w··.···* ··Y···6a
00b0  08 ae be 1b 61 d1 94 08  ee 1f 6a 64 35 79 8c 22   ····a··· ··jd5y·"
00c0  a9 35 d8 7a 46 10 4f 87  22 67 9b d1 c5 f2 b6 16   ·5·zF·O· "g·····
```

Large Scale Evaluation
Testing for Weak STEK




```
▾ Handshake Protocol: New Session Ticket
    Handshake Type: New Session Ticket (4)
    Length: 235
  ▾ TLS Session Ticket
      Session Ticket Lifetime Hint: 100800 seconds (1 day, 4 hours)
      Session Ticket Length: 229
      Session Ticket: 02d32b3f673ba516106107535a8f78ad158f3134dca563e7…
▸ TLSv1.2 Record Layer: Change Cipher Spec Protocol: Change Cipher Spec
▾ TLSv1.2 Record Layer: Handshake Protocol: Encrypted Handshake Message
    Content Type: Handshake (22)
    Version: TLS 1.2 (0x0303)
```

```
0050  c0 00 e5 02 d3 2b 3f 67  3b a5 16 10 61 07 53 5a   ···· ··+?g ;···a·SZ
0060  8f 78 ad 15 8f 31 34 dc  a5 63 e7 70 31 27 6d 73   ·x···14· ·c·p1'ms
0070  13 1c 6a 0c 09 bc b9 71  41 fd 60 fc 1a 4b dd 7d   ··j····q A·`··K·}
0080  6d ac 0d 59 cb 45 48 c6  3e b0 02 3e 4f e7 f7 45   m··Y·EH· >··>O··E
0090  73 28 eb 73 93 58 7a c0  68 f7 c6 d4 d0 20 66 a7   s(·s·Xz· h···· f·
00a0  77 8d 97 2e 0f f4 e6 2a  d7 a3 59 db b2 a2 36 61   w··.···* ··Y···6a
00b0  08 ae be 1b 61 d1 94 08  ee 1f 6a 64 35 79 8c 22   ····a··· ··jd5y·"
00c0  a9 35 d8 7a 46 10 4f 87  22 67 9b d1 c5 f2 b6 16   ·5·zF·O· "g·····
```

Large Scale Evaluation
Testing for Weak STEK

1. Decrypt Ticket
2. Check for master secret 🔑 in plaintext

- Unknown algorithm, key, IV, structure

```
b26b38b49fa971dee53e5137b0e73226c74e0cbd21d5c4042fad0b0a8c6747ef5e43969a42ef883b1c529fc
```

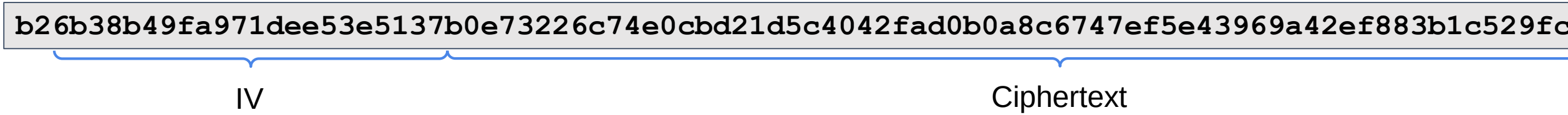
IV

Ciphertext

Large Scale Evaluation
Testing for Weak STEK

1. Decrypt Ticket
2. Check for master secret 🔑 in plaintext

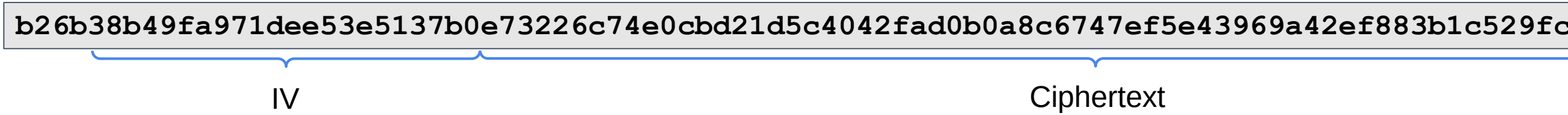
- Unknown algorithm, key, IV, structure



Large Scale Evaluation
Testing for Weak STEK

1. Decrypt Ticket
2. Check for master secret 🔑 in plaintext

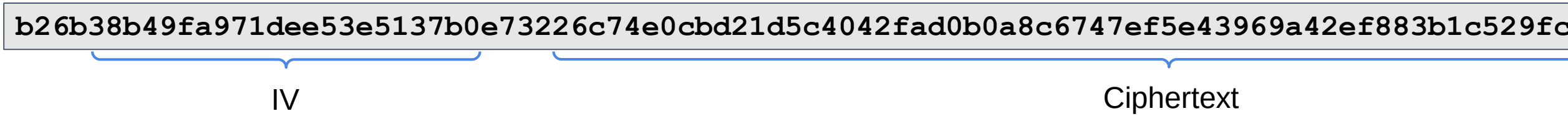
- Unknown algorithm, key, IV, structure



Large Scale Evaluation
Testing for Weak STEK

1. Decrypt Ticket
2. Check for master secret 🔑 in plaintext

- Unknown algorithm, key, IV, structure



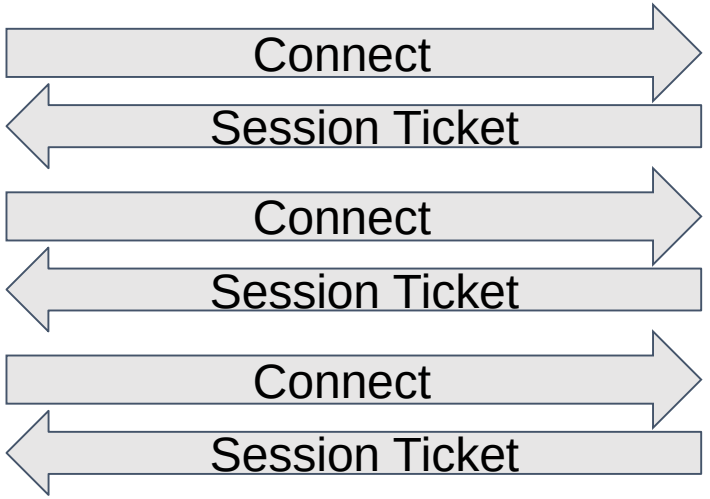
- AES, ChaCha, DES, 3DES?
  - ECB, CBC, CTR, GCM?
- Key?
  - 0x000000, 0x000102, 0xFFFFFF

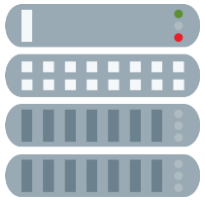
Large Scale Evaluation
Brute Forcing the STEK

- Tested TLS “related” encryption algorithm combinations:
  - DES : ECB, CBC mode
  - 3DES : ECB, CBC mode
  - AES-128/256: ECB, CBC, CTR, CCM, GCM mode (ignoring auth)
  - ChaCha20
- Tested TLS “related” authentication algorithms:
  - HMAC: MD5, SHA1, SHA256, SHA384 and SHA512
- 144 potential keys
- >400 potential formats (encryption)

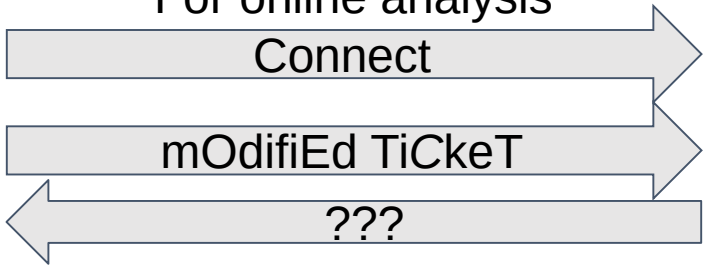
Large Scale Evaluation Methodology

10x TLS 1.3
10x TLS 1.2
…

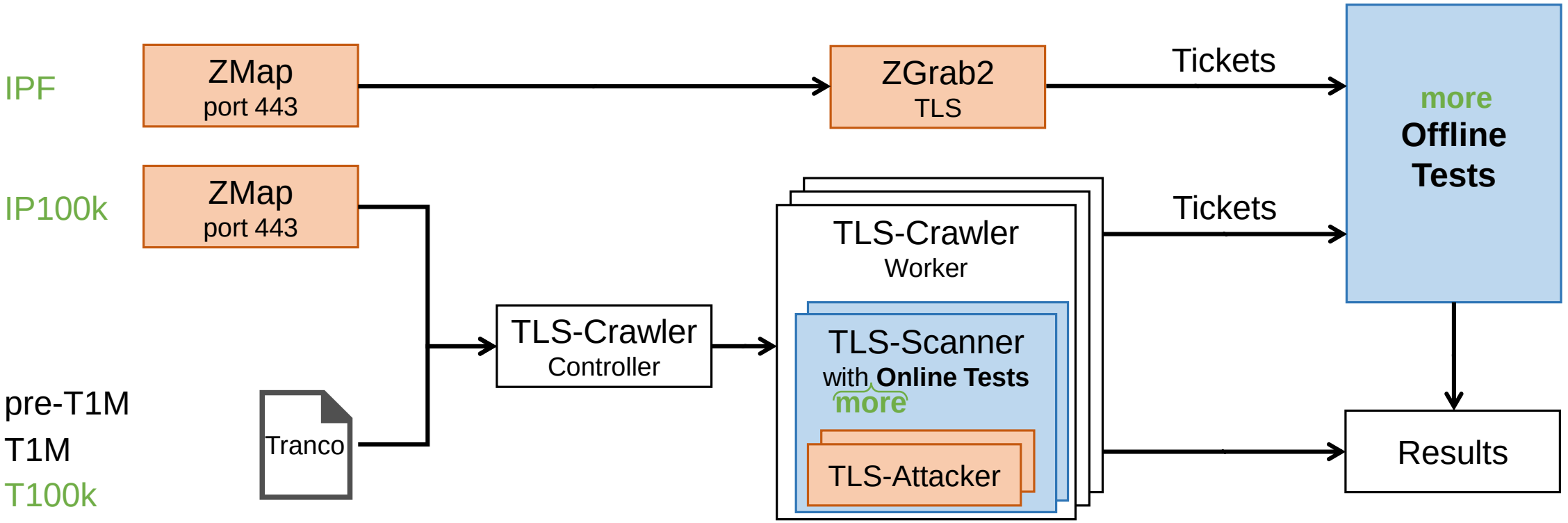




For online analysis



Scans


IPF
ZMap
port 443
ZGrab2
TLS
Tickets
more
Offline
Tests
IP100k
ZMap
port 443
TLS-Crawler
Worker
Tickets
TLS-Crawler
Controller
TLS-Scanner
with Online Tests
more
pre-T1M
T1M
T100k
Tranco
TLS-Attacker
Results

Results

	Offline Analysis			Online Analysis	
Scan	**Unencrypted Ticket**	**Weak STEK**	**Reused Keystream**	**Missing Auth. Protection**	**Padding Oracle**
pre-T1M	0	1923	–	–	–

- Scope: Tranco 100k – **1.9%** vulnerable
- Most of the servers belonged to AWS
- STEK = 0x0000….
- Reported April 2021, fixed within 8 hours
- According to AWS, bug was maybe introduced in September 2020
- Probable cause: Change in the STEK format of NGINX caused wrong “offsets”

Results

		Offline Analysis		Online Analysis	
Scan	**Unencrypted Ticket**	**Weak STEK**	**Reused Keystream**	**Missing Auth. Protection**	**Padding Oracle**
pre-T1M	0	1923	–	–	–
T1M	0				
T100k	0				
IP100k	0				
IPF	0				

Results

	Offline Analysis			Online Analysis	
Scan	Unencrypted Ticket	Weak STEK	**Reused Keystream**	Missing Auth. Protection	Padding Oracle
pre-T1M	0	1923	–	–	–
T1M	0		–		
T100k	0		0		
IP100k	0		0		
IPF	0		1		

Results

	Offline Analysis			Online Analysis	
Scan	**Unencrypted Ticket**	**Weak STEK**	**Reused Keystream**	**Missing Auth. Protection**	**Padding Oracle**
pre-T1M	0	1923	–	–	–
T1M	0		–	–	–
T100k	0		0	0	0
IP100k	0		0	0	0
IPF	0		1	–	–

Results

	Offline Analysis			Online Analysis	
Scan	Unencrypted Ticket	**Weak STEK**	Reused Keystream	Missing Auth. Protection	Padding Oracle
pre-T1M	0	1923	–	–	–
T1M	0	3	–	–	–
T100k	0	1	0	0	0
IP100k	0	0	0	0	0
IPF	0	189	1	–	–

Results
Weak Keys

	Servers Found	Encryption		Authentication	
		Algorithm	Key	Algorithm	Key
mostly AWS	1908	AES-256-CBC	`00 00 ... 00 00`	-	–

Results
Weak Keys

	Servers Found	Encryption		Authentication	
		Algorithm	Key	Algorithm	Key
mostly AWS	1908	AES-256-CBC	00 00 ... 00 00	-	–
	118	AES-128-CBC	00 00 ... 00 00	HMAC-SHA256	00 00 ... 00 00

Results
Weak Keys

	Servers Found	Encryption		Authentication	
		Algorithm	Key	Algorithm	Key
mostly AWS	1908	AES-256-CBC	00 00 ... 00 00	-	–
	118	AES-128-CBC	00 00 ... 00 00	HMAC-SHA256	00 00 ... 00 00
	12	AES-256-CBC	00 00 ... 00 00	HMAC-SHA384	00 00 ... 00 00

Results
Weak Keys

	Servers Found	Encryption		Authentication	
		Algorithm	Key	Algorithm	Key
mostly AWS	1908	AES-256-CBC	00 00 ... 00 00	-	–
	118	AES-128-CBC	00 00 ... 00 00	HMAC-SHA256	00 00 ... 00 00
	12	AES-256-CBC	00 00 ... 00 00	HMAC-SHA384	00 00 ... 00 00
	3				

aes key	10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f
hmac key	20 21 22 23 24 25 26 27 28 29 2a 2b 2c 2d 2e 2f 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Results
Weak Keys

Servers Found	Encryption		Authentication	
	Algorithm	Key	Algorithm	Key
mostly AWS 1908	AES-256-CBC	00 00 ... 00 00	-	–
118	AES-128-CBC	00 00 ... 00 00	HMAC-SHA256	00 00 ... 00 00
12	AES-256-CBC	00 00 ... 00 00	HMAC-SHA384	00 00 ... 00 00
3				

key name	00	01	02	03	04	05	06	07	08	09	0a	0b	0c	0d	0e	0f
aes key	10	11	12	13	14	15	16	17	18	19	1a	1b	1c	1d	1e	1f
hmac key	20	21	22	23	24	25	26	27	28	29	2a	2b	2c	2d	2e	2f
	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

Results
Weak Keys

Servers Found	Encryption		Authentication	
	Algorithm	Key	Algorithm	Key
mostly AWS 1908	AES-256-CBC	00 00 ... 00 00	-	–
118	AES-128-CBC	00 00 ... 00 00	HMAC-SHA256	00 00 ... 00 00
12	AES-256-CBC	00 00 ... 00 00	HMAC-SHA384	00 00 ... 00 00
3	AES-128-CBC	10 11 ... 1e 1f	HMAC-SHA256	20...2f 00...00

key name	00	01	02	03	04	05	06	07	08	09	0a	0b	0c	0d	0e	0f
aes key	10	11	12	13	14	15	16	17	18	19	1a	1b	1c	1d	1e	1f
hmac key	20	21	22	23	24	25	26	27	28	29	2a	2b	2c	2d	2e	2f
	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

Results
Weak Keys

Servers Found	Encryption		Authentication	
	Algorithm	Key	Algorithm	Key
mostly AWS 1908	AES-256-CBC	00 00 ... 00 00	-	–
118	AES-128-CBC	00 00 ... 00 00	HMAC-SHA256	00 00 ... 00 00
12	AES-256-CBC	00 00 ... 00 00	HMAC-SHA384	00 00 ... 00 00
3	AES-128-CBC	10 11 ... 1e 1f	HMAC-SHA256	20...2f 00...00
75				

aes key	31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
hmac key	31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Results
Weak Keys

Servers Found	Encryption		Authentication	
	Algorithm	Key	Algorithm	Key
mostly AWS 1908	AES-256-CBC	00 00 ... 00 00	-	–
118	AES-128-CBC	00 00 ... 00 00	HMAC-SHA256	00 00 ... 00 00
12	AES-256-CBC	00 00 ... 00 00	HMAC-SHA384	00 00 ... 00 00
3	AES-128-CBC	10 11 ... 1e 1f	HMAC-SHA256	20...2f 00...00
75	AES-256-CBC	31...31 00...00	HMAC-SHA256	31...31 00...00

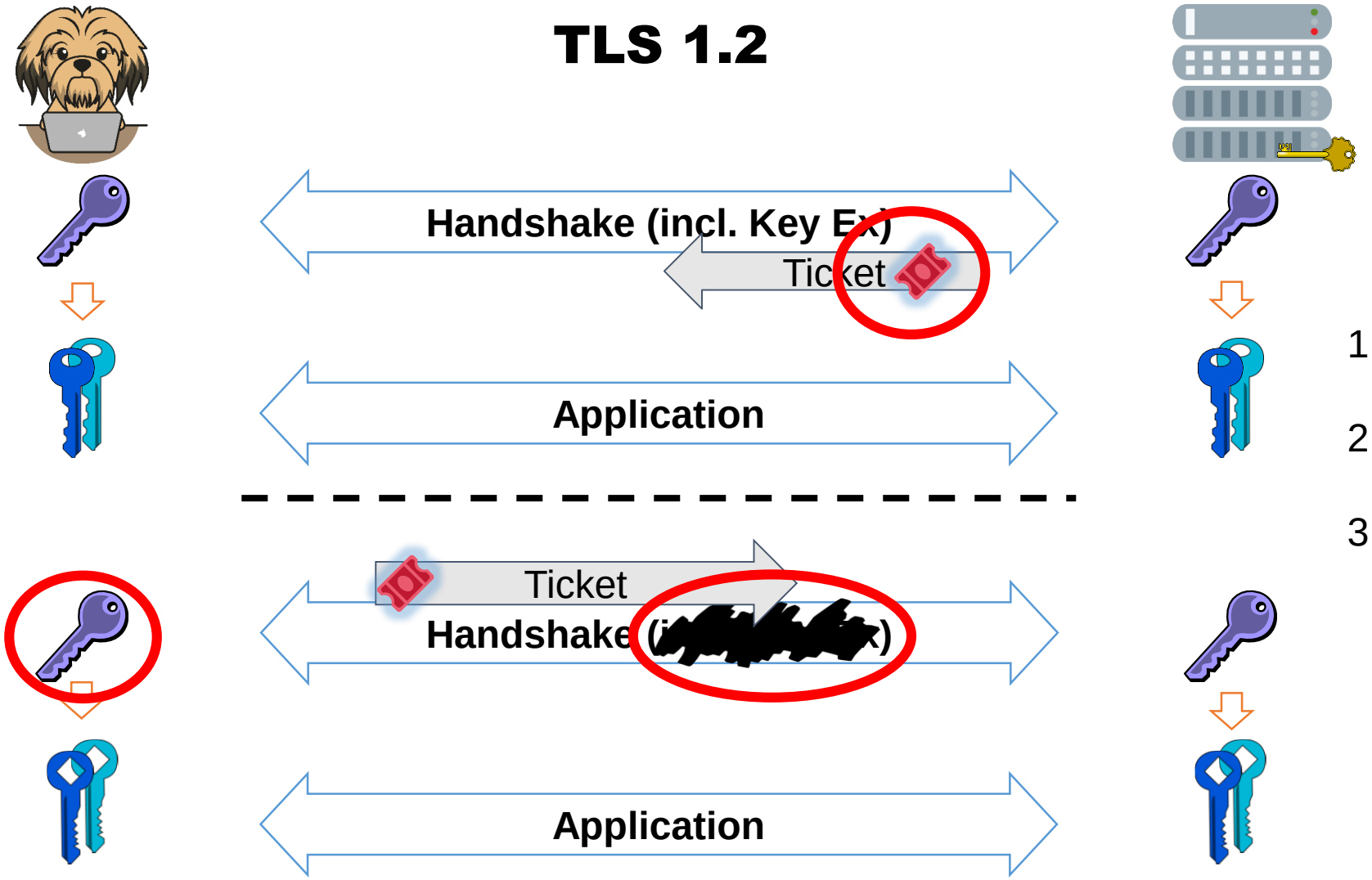
aes key	31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
hmac key	31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Results
Summary

- No authentication issues
- One reused keystream
- Weak keys
  - Many 0-keys
  - 00 01 02 03
  - Partially initialized

We *Really* Need to Talk About TLS Session Tickets

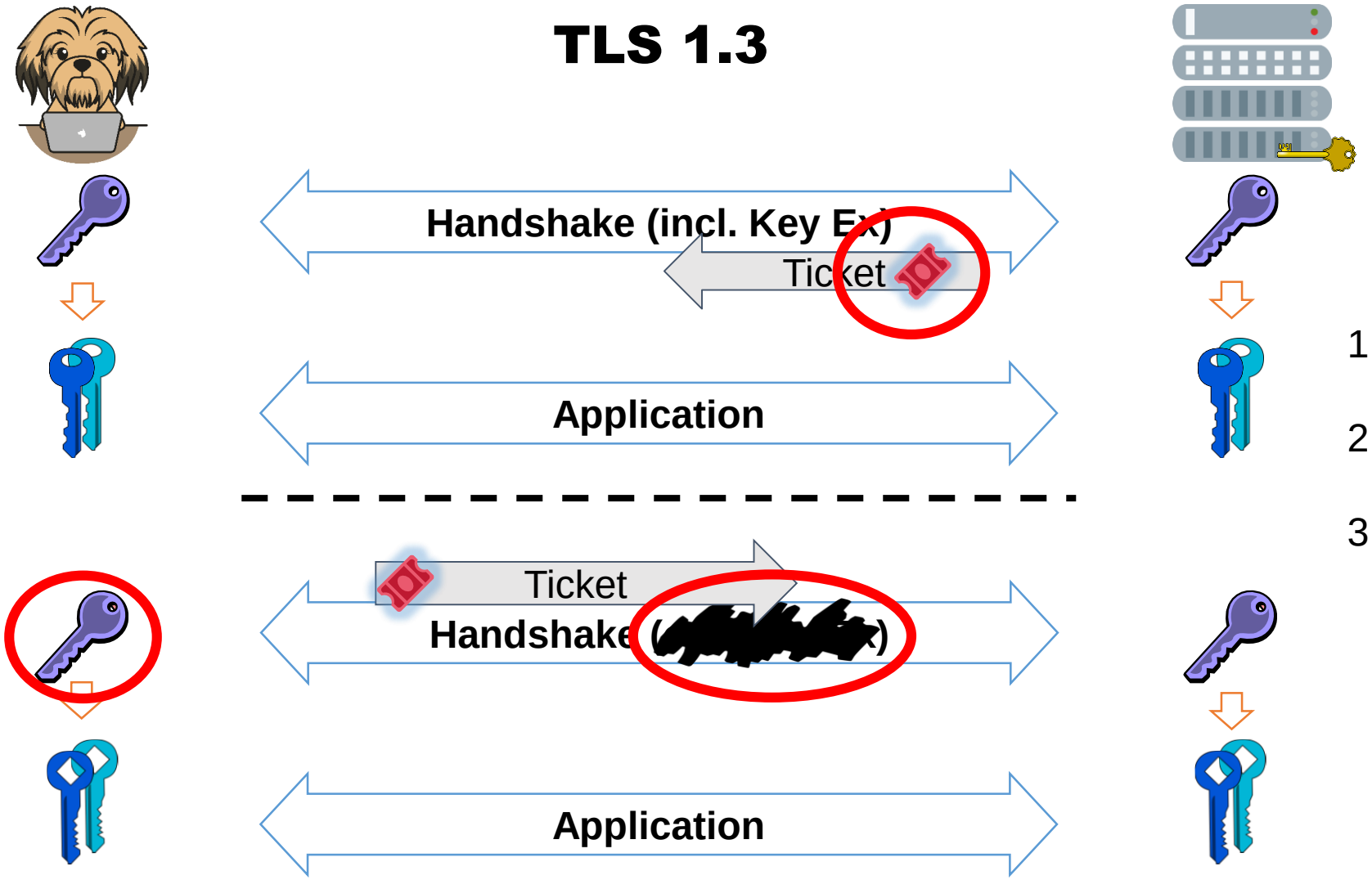
TLS 1.2



1. Can decrypt recorded sessions
2. Same secret reused
3. Tickets sent in plaintext

We *Really* Need to Talk About TLS Session Tickets

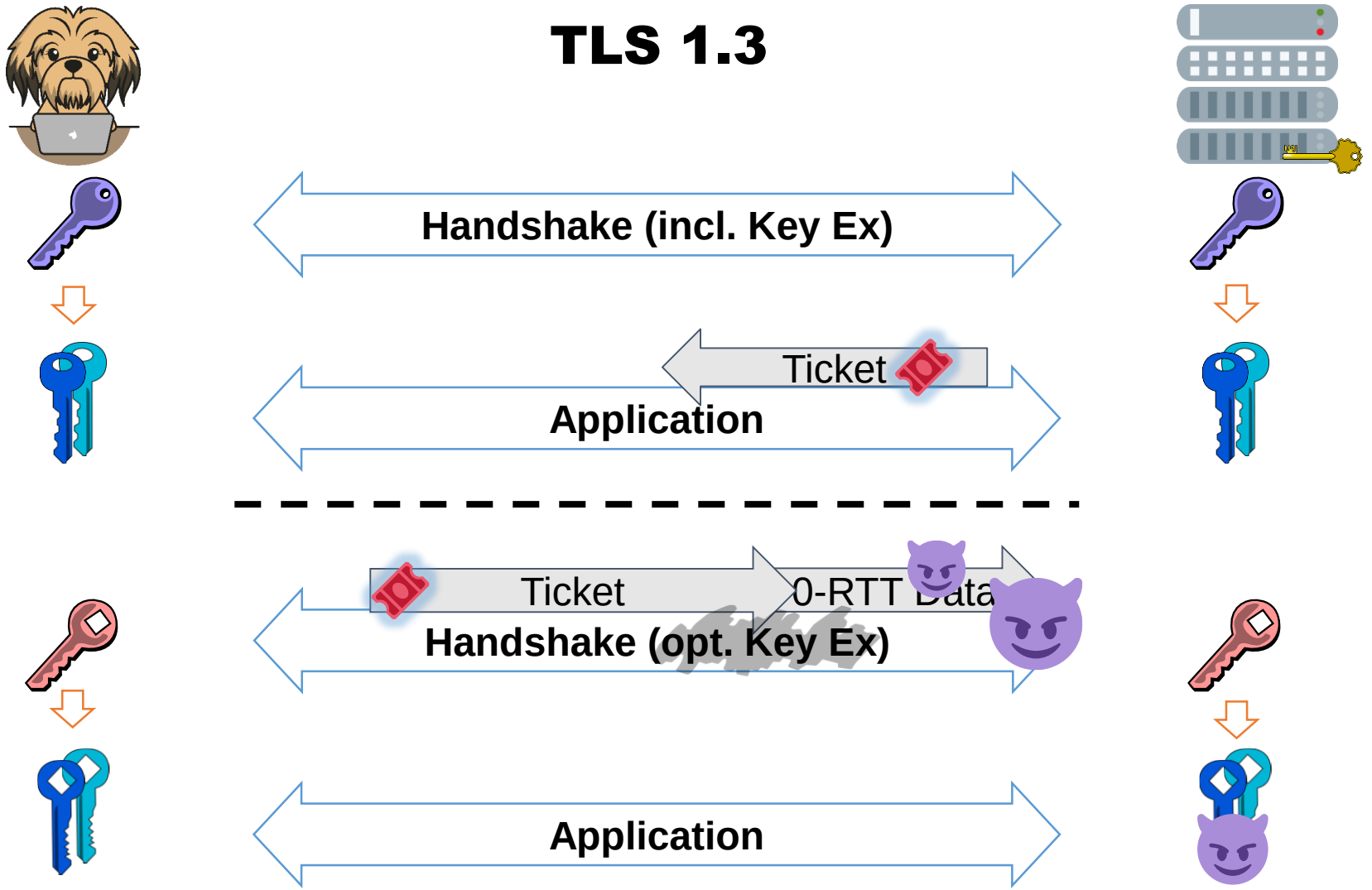
TLS 1.3



1. Can decrypt recorded sessions
   - Allow key exchange
2. Same secret reused
   - Derive new secrets
3. Tickets sent in plaintext
   - Sent encrypted

We *Really* Need to Talk About TLS Session Tickets

TLS 1.3



- If no Key Ex:
  - Decrypt Application
- Read 0-RTT Data
- Impersonate Server
- TLS 1.2 is widely used

We Really Need to Talk About Session Tickets

We Really Need(ed) to Talk About Session Tickets

Findings

- 0000 isn’t a secure key
- Tickets undermine TLS security guarantees

Conclusions

- Hidden danger in
  - Crypto shortcuts
  - Silently breaking crypto
  - Unverifiable crypto

Takeaways

- Design protocols verifiable for both parties
- Add defense in depth to your implementation
  - Check key material before use

We Really Need(ed) to Talk About Session Tickets

Findings

- 0000 isn’t a secure key
- Tickets undermine TLS security guarantees

Conclusions

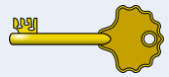
- Hidden danger in
  - Crypto shortcuts
  - Silently breaking crypto
  - Unverifiable crypto

Questions?

Takeaways

- Design protocols verifiable for both parties
- Add defense in depth to your implementation
  - Check key material before use

TLS Keys

	Key Ex. Key 	Master Secret 	Encryption Key 	MAC Key 	Certificate Key 	CA Certificate Key 	STEK 
How many Users/Servers	One User	One User	One User	One User	All Users of One Server	All Users	All Users of One Server
Passive Impact	Decrypt Traffic	Decrypt Traffic	Decrypt Traffic	-	Maybe Key Exchange Key	-	Decrypt Traffic
Active Impact	Intercept Traffic	Intercept Traffic	Intercept Traffic	Alter messages	Impersonate Server	Impersonate Server	Impersonate Server
Validity	One Session (+Resumption)	One Connection	One Connection	One Connection	Months-Years	Years	Hours-Weeks
Externally Auditable	Partially Doable	(random)	(random)	(random)	Yes	Yes	Hard

TLS Keys

	Key Ex. Key	Master Secret	Encryption Key	MAC Key	Certificate Key	CA Certificate Key	STEK
How many U…	One User	One User	One User				
P…			Decrypt Traffic				…ffic
A…			Intercept Traffic				…e
V…	(+Resumption)	Connection	One Connection	Co…			…cks
Externally Auditable	Partially Doable	(random)	(random)	(random)	Yes	Yes	Hard

Measuring small subgroup attacks against Diffie-Hellman

Luke Valenta*, David Adrian†, Antonio Sanso‡, Shaanan Cohney*,
Joshua Fried*, Marcella Hastings*, J. Alex Halderman†, Nadia Heninger*
*University of Pennsylvania
†University of Michigan
‡Adobe

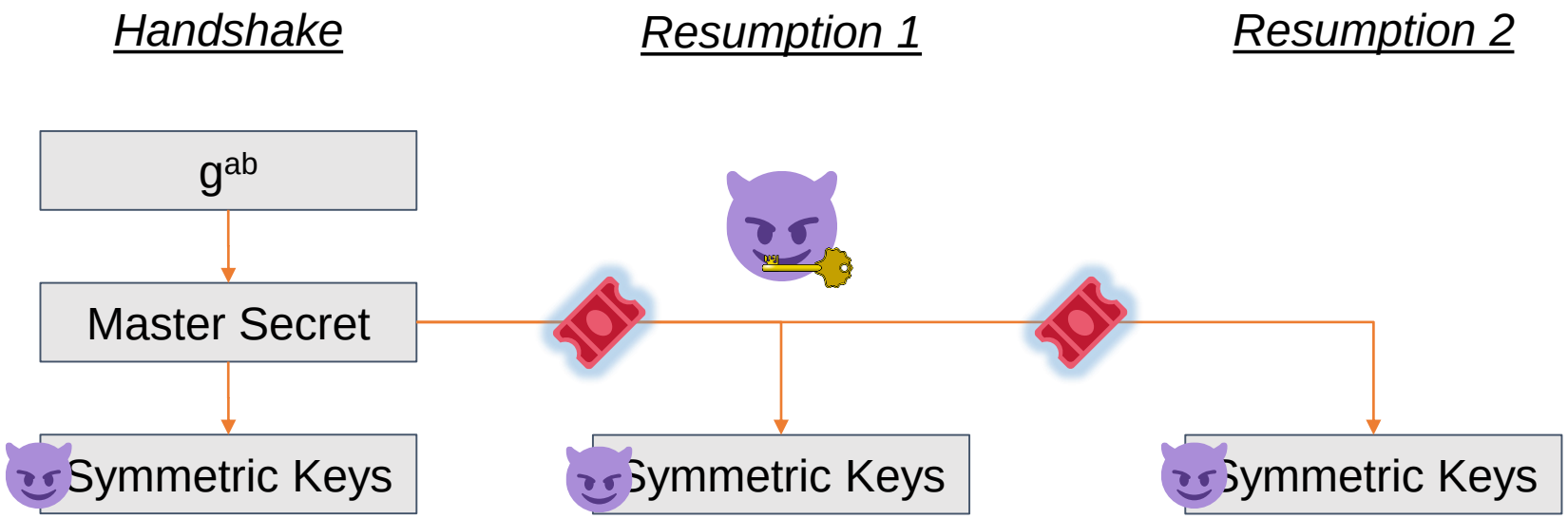
Mining Your Ps and Qs: Detection of Widespread Weak Keys in Network Devices

Nadia Heninger†* Zakir Durumeric‡* Eric Wustrow‡ J. Alex Halderman‡

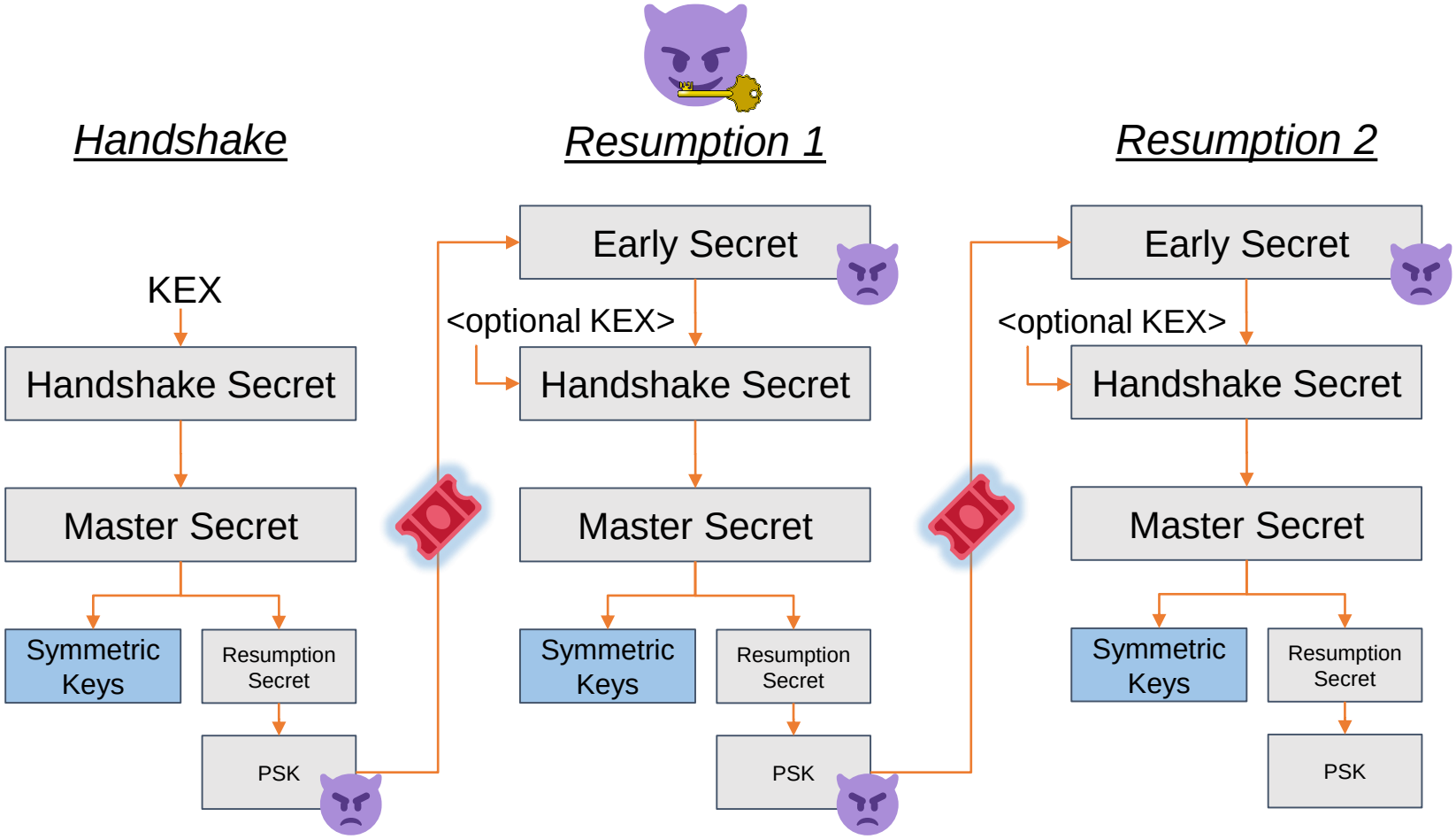
badkeys.info

Checking cryptographic public keys for known vulnerabilities

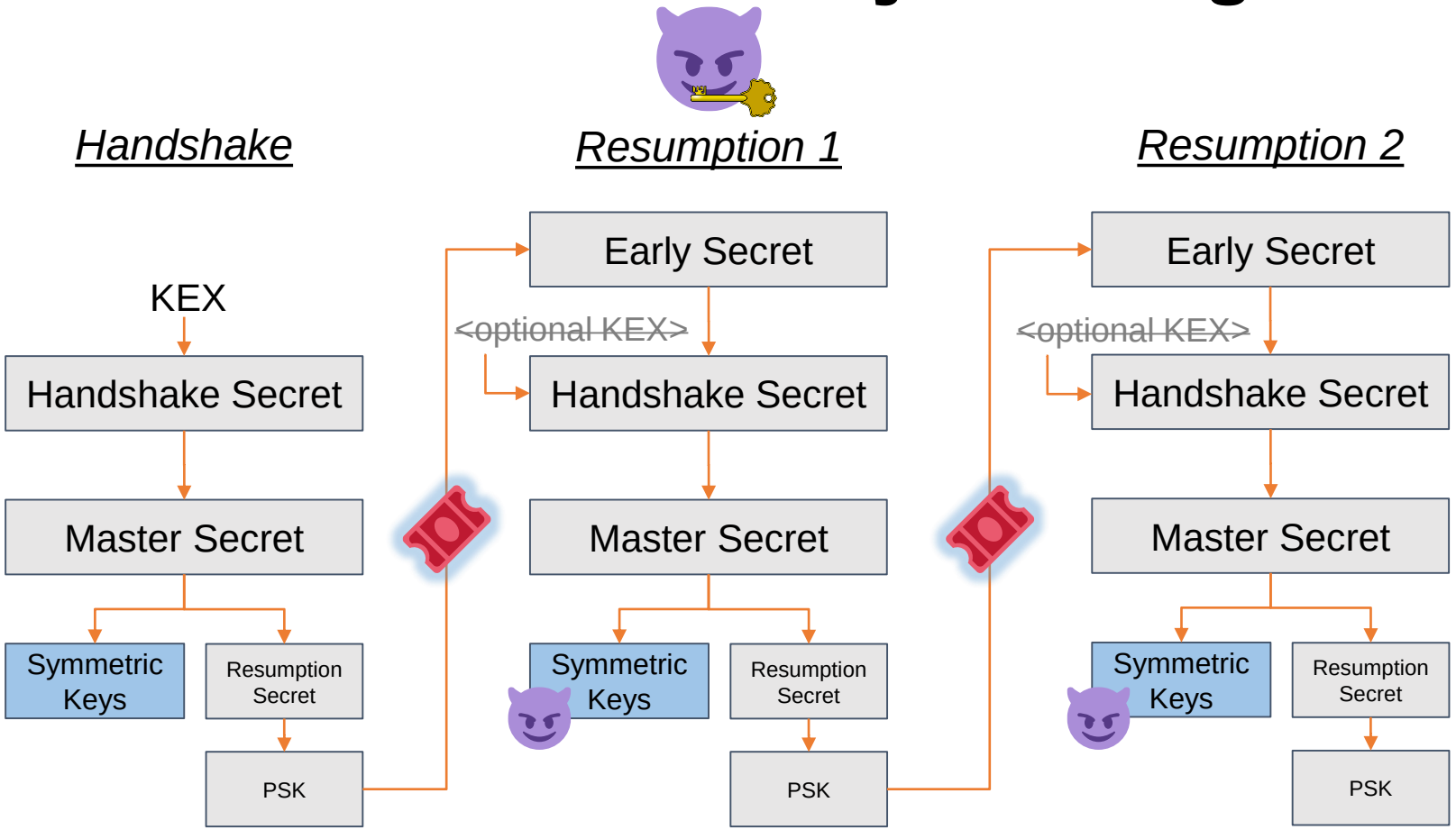
Key Derivation
TLS 1.2



Key Derivation
TLS 1.3



Key Derivation
TLS 1.3 - Without Key Exchange



Tested Keys

Repeating bytes

- **00000000**…
- **01010101**…
- **0F0F0F0F**…
- **10101010**…
- **FFFFFFFF**…

…

48 keys

Increasing bytes

- **00010203**…
- **10111213**…
- **20212223**…
- **30313233**…
- **40414243**…

…

48 keys

Stepping bytes

- **00102030**…
- **00112233**…
- **00011223**…

3 keys

Known Magic Constants (wikipedia)

- **ABADCAFE**…
- **DEADBEEF**…
- **DEADBABE**…
- **BAAAAAAD**…
- **BAD22222**…
- **BADBADBA**…
- **CAFEFEED**…

…

47 keys

NIST Example Keys

- **2b7e151628aed2a6abf7158809cf4**…
- **8e73b0f7da0e6452c810f32b80907**…
- **603deb1015ca71be2b73aef0857d7**…

3 keys

144 keys total

encrypted_state

```
struct {
    ProtocolVersion protocol_version;
    CipherSuite cipher_suite;
    CompressionMethod compression_method;
    opaque master_secret[48];
    ClientIdentity client_identity;
    uint32 timestamp;
} StatePlaintext;
```

TLS Handshake

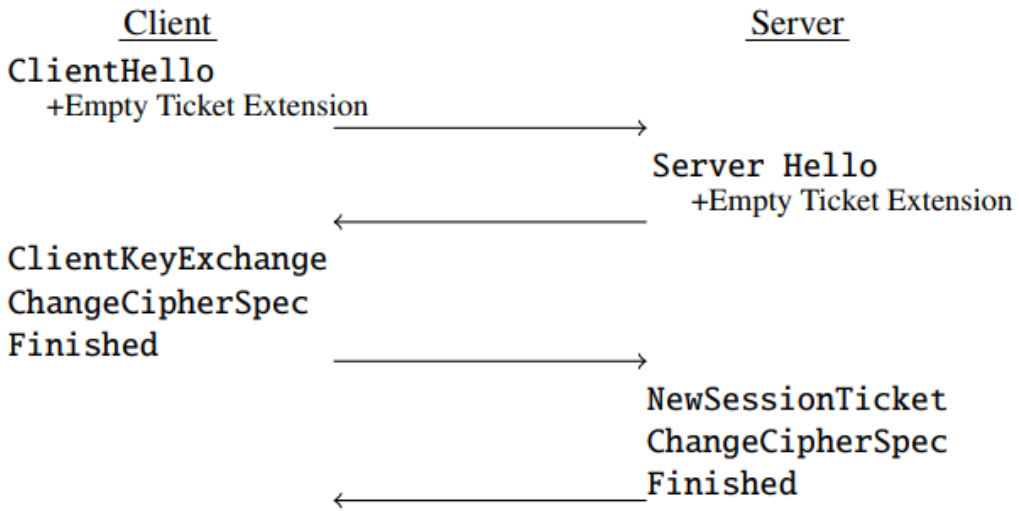


Figure 1: A TLS 1.2 handshake using a Diffie-Hellman key exchange and session ticket negotiation.

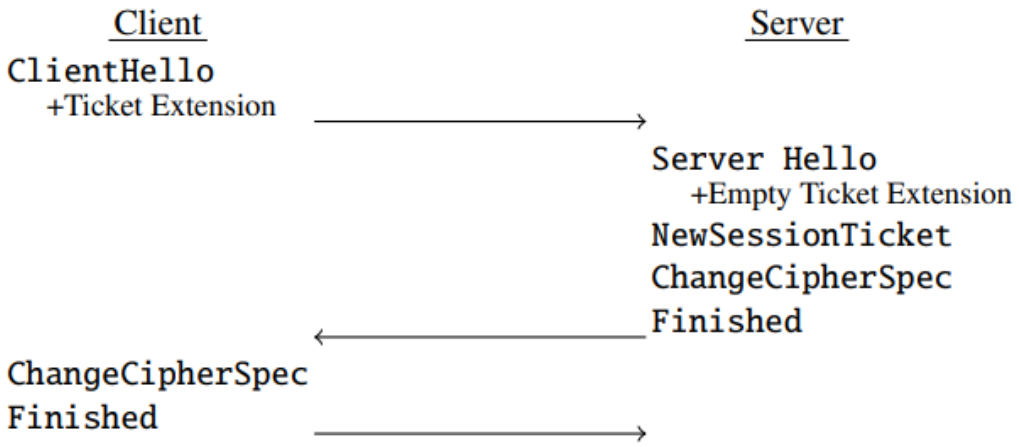


Figure 2: A TLS 1.2 resumption handshake using session tickets

Libraries

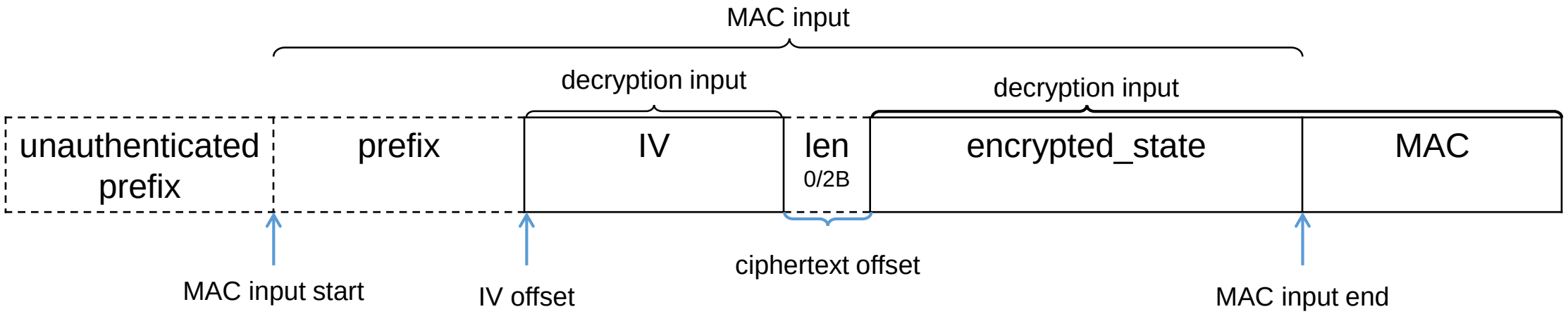
		Session Ticket Format						Symmetric Algorithms	
Library	**Version**	`magic`[a]	`key_name`	`seed`[a]	`iv`[b]	`len`	`mac`	**Encryption**	**Authentication**
RFC 5077		–	**16**	–	**16**	**2**	**32**	**AES-128-CBC**	**HMAC-SHA256**
BoringSSL	2021[c]	–	**16**	–	**16**	–	**32**	**AES-128-CBC**	**HMAC-SHA256**
Botan	2.19.2	8	4	16	12	–	16	**AES**-256-GCM	(GMAC)
GnuTLS	3.7.6	–	**16**	–	**16**	**2**	20	**AES**-256-**CBC**	**HMAC**-SHA1
GoTls	go1.18.3	–	**16**	–	**16**	–	**32**	**AES-128**-CTR	**HMAC-SHA256**
MatrixSSL (TLS 1.2)	4.3.0	–	**16**	–	**16**	–	**32**	**AES**-256-**CBC**	**HMAC-SHA256**
MatrixSSL (TLS 1.3)	4.3.0	–	**16**	–	12	–	16	**AES**-256-GCM	(GMAC)
mbedTLS[d]	3.1.0	–	4	–	12	**2**	16	**AES-128**/256-GCM **AES-128**/256-CCM	(GMAC) (CBCMAC)
OpenSSL	3.0.3	–	**16**	–	**16**	–	**32**	**AES**-256-**CBC**	**HMAC-SHA256**
Rustls	0.20.6	–	–	–	12	–	16	ChaCha20	Poly1305
s2n	1.3.15	–	**16**	–	12	–	16	**AES**-256-GCM	(GMAC)
Apache	2.4.54	*Format of OpenSSL*						**AES-128-CBC**	**HMAC-SHA256**
Nginx	1.22.0	*Format of OpenSSL*						**AES-128**/256-**CBC**	**HMAC-SHA256**
OpenLiteSpeed	1.17.6	*Format of BoringSSL*						**AES-128-CBC**	**HMAC-SHA256**

a: These fields are only added by Botan.
b: IV or Nonce.
c: BoringSSL does not use releases. We analyzed the commit `dddb60e` from 2021-08-31.
d: mbedTLS can be configured to use different algorithms.

Assumed Format

Encryption: 434 Formats

Authentication: 126 Formats



Full Results

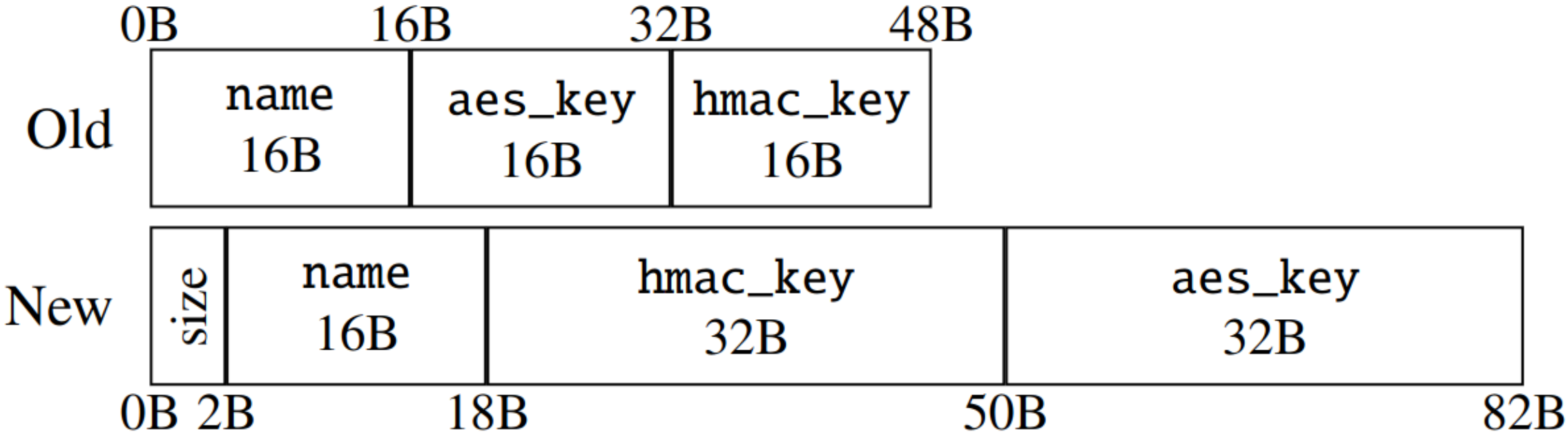
			Statistics			Offline Analysis			Online Analysis	
Scan	**Date**	**Tested Versions**	**Supports TLS**	**Issues Ticket**	**Resumes Ticket**	**Unencrypted Ticket**	**Weak STEK**	**Reused Keystream**	**Missing Auth. Protection**	**Padding Oracle**
pre-T1M	2021-04	1.2	66,992	53,059	–	0	1,923	–	–	–
T1M	2021-05	1.2 – 1.3	760,293	594,238	547,159	0	3	–	–	–
T100k	2022-04	1.0 – 1.3	71,200	58,069	55,003	0	1	0	0	0
IP100k	2022-04	1.0 – 1.3	80,972	57,493	55,969	0	0	0	0	0
IPF	2022-08	≤1.2	39,390,365	29,621,531	–	0	189	1	–	–

Found Keys

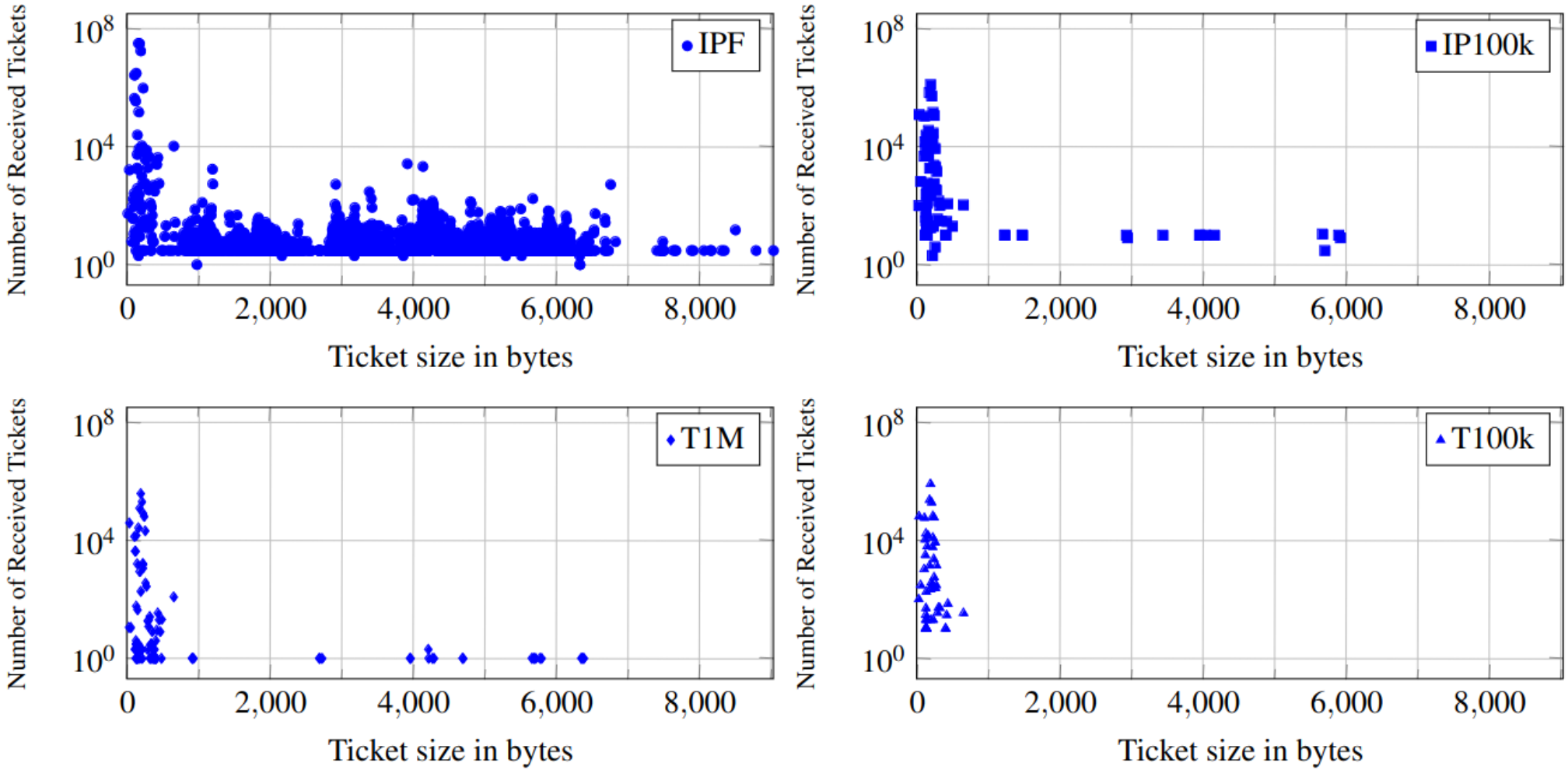
Scan	Servers Found	Encryption		Authentication	
		Algorithm	Key	Algorithm	Key
pre-T1M	1903	AES-256-CBC	00 00 ... 00 00	–	–
	20	AES-128-CBC	00 00 ... 00 00	HMAC-SHA256	00 00 ... 00 00
T1M	3	AES-128-CBC	00 00 ... 00 00	HMAC-SHA256	00 00 ... 00 00
T100k	1	AES-128-CBC	00 00 ... 00 00	HMAC-SHA256	00 00 ... 00 00
IPF	5	AES-256-CBC	00 00 ... 00 00	–	–
	94	AES-128-CBC	00 00 ... 00 00	HMAC-SHA256	00 00 ... 00 00
	12	AES-256-CBC	00 00 ... 00 00	HMAC-SHA384	00 00 ... 00 00
	3	AES-128-CBC	10 11 ... 1e 1f	HMAC-SHA256	20...2f 00...00[a]
	75	AES-256-CBC	31...31 00...00[b]	HMAC-SHA256	31...31 00...00[b]

a: This key consists of 16 consecutively increasing bytes followed by 16 0x00 bytes.
b: This key consists of 16 0x31 bytes followed by 16 0x00 bytes.

nginx change



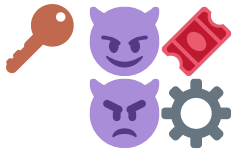
Ticket Size


Number of Received Tickets
10^8
10^4
10^0
0
2,000
4,000
6,000
8,000
Ticket size in bytes
IPF
IP100k
T1M
T100k

Attributions

Based on MesserWoland, Crypto key, CC BY-SA 3.0
Path and Alignment slightly adapted, Colors changed for some figures

Based on https://publicdomainvectors.org/en/free-clipart/Vector-drawing-of-grayscale-key/31029.html



twemoji https://github.com/twitter/twemoji

https://publicdomainvectors.org/en/free-clipart/Vector-image-of-old-style-decorative-door-key/21178.html

https://gitlab.com/rossel.jost/latex-twemojis/-/tree/master/src/twemojis-extra/