

On the Practical (In-)Security of 64-bit Block Ciphers

Collision Attacks on HTTP over TLS and OpenVPN

Karthikeyan Bhargavan Gaëtan Leurent

Inria, France

ACM CCS 2016



Choosing a cipher

What are the main factors determining the security of a cipher?

Cipher	Key-length	Cryptanalysis	Block size
DES	56 bits	2^{40}	64 bits
3DES	168 bits	2^{112}	64 bits
Blowfish	32–448 bits	None	64 bits
RC4	40–2048 bits	2^8	stream
AES	128–256 bits	Related-key	128 bits

Main point of the talk

- ▶ Block size does matter
- ▶ Practical attacks against 64-bit block ciphers



Choosing a cipher

What are the main factors determining the security of a cipher?

Cipher	Key-length		Cryptanalysis	Block size
DES	56 bits	✗	2^{40}	64 bits
3DES	168 bits	✓	2^{112}	64 bits
Blowfish	32–448 bits	✓	None	64 bits
RC4	40–2048 bits	✓	2^8	stream
AES	128–256 bits	✓	Related-key	128 bits

Main point of the talk

- ▶ Block size does matter
- ▶ Practical attacks against 64-bit block ciphers



Choosing a cipher

What are the main factors determining the security of a cipher?

Cipher	Key-length		Cryptanalysis		Block size
DES	56 bits	✗	2^{40}	✗	64 bits
3DES	168 bits	✓	2^{112}	✓	64 bits
Blowfish	32–448 bits	✓	None	✓	64 bits
RC4	40–2048 bits	✓	2^8	✗	stream
AES	128–256 bits	✓	Related-key	✓	128 bits

Main point of the talk

- ▶ Block size does matter
- ▶ Practical attacks against 64-bit block ciphers



Choosing a cipher

What are the main factors determining the security of a cipher?

Cipher	Key-length		Cryptanalysis		Block size	
DES	56 bits	✗	2^{40}	✗	64 bits	✗
3DES	168 bits	✓	2^{112}	✓	64 bits	✗
Blowfish	32–448 bits	✓	None	✓	64 bits	✗
RC4	40–2048 bits	✓	2^8	✗	stream	✓
AES	128–256 bits	✓	Related-key	✓	128 bits	✓

Main point of the talk

- ▶ Block size does matter
- ▶ Practical attacks against 64-bit block ciphers



Choosing a cipher

What are the main factors determining the security of a cipher?

Cipher	Key-length		Cryptanalysis		Block size	
DES	56 bits	✗	2^{40}	✗	64 bits	✗
3DES	168 bits	✓	2^{112}	✓	64 bits	✗
Blowfish	32–448 bits	✓	None	✓	64 bits	✗
RC4	40–2048 bits	✓	2^8	✗	stream	✓
AES	128–256 bits	✓	Related-key	✓	128 bits	✓

Main point of the talk

- ▶ Block size does matter
- ▶ Practical attacks against 64-bit block ciphers

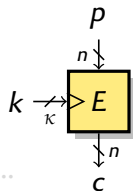


Block ciphers and Modes of operation

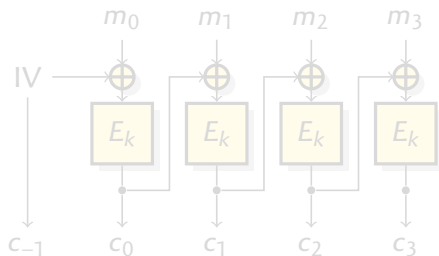
- ▶ A block cipher is a **family of permutations**:

$$\{0, 1\}^{\kappa}, \{0, 1\}^n \rightarrow \{0, 1\}^n$$

$$k, p \mapsto c$$



- ▶ It is used with a **mode of operation**: CBC, CTR, GCM, ...
 - ▶ To deal with variable-length messages
 - ▶ To include randomness
 - ▶ Important example: CBC

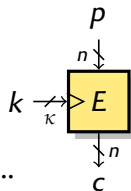


Block ciphers and Modes of operation

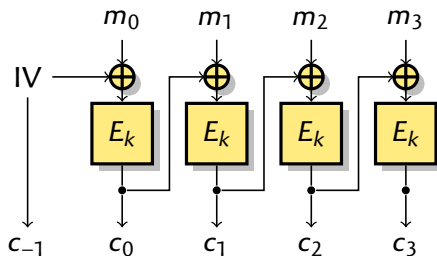
- ▶ A block cipher is a **family of permutations**:

$$\{0, 1\}^\kappa, \{0, 1\}^n \rightarrow \{0, 1\}^n$$

$$k, p \mapsto c$$

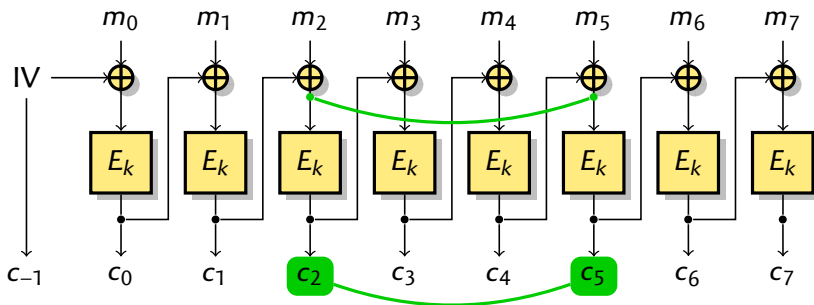


- ▶ It is used with a **mode of operation**: CBC, CTR, GCM, ...
 - ▶ To deal with variable-length messages
 - ▶ To include randomness
 - ▶ Important example: CBC



CBC collisions

- ▶ Security of modes can be lower than security of cipher
- ▶ Well known collision attack against CBC



- ▶ If $c_i = c_j$, then $c_{i-1} \oplus m_i = c_{j-1} \oplus m_j$
- ▶ Ciphertext collision reveals the **xor of two plaintext blocks**

Birthday paradox

The birthday paradox

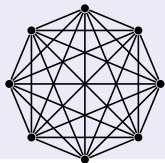
In a room with 23 people, there is a 50% chance that two of them share the same birthday.



Birthday attack

When drawing random n -bit strings, a collision is expected after roughly $2^{n/2}$ draws.

More generally, 2^{2t-n} collisions with 2^t draws



- ▶ CBC leaks plaintext after $2^{n/2}$ blocks encrypted with the same key
 - ▶ In a single message or many different messages

Security of modes of operation

- ▶ Modes are proven secure assuming the block cipher is secure.
- ▶ Most modes (CBC, CTR, GCM, ...) have a security proof like:

$$\text{Adv}_{\text{CBC-}E}^{\text{CPA}}(q, t) \leq \text{Adv}_E^{\text{PRP}}(q', t') + \frac{\sigma^2}{2^n}$$

- ▶ The CPA security of CBC is essentially the PRP security of E (the block cipher)
- ▶ As long as the number of encrypted blocks $\sigma \lll 2^{n/2}$
 - ▶ Usually matching attack with birthday complexity ($2^{n/2}$)
 - ▶ With a 64-bit cipher, the bound is **only 32 GB**

Security of modes of operation

- ▶ Modes are proven secure assuming the block cipher is secure.
- ▶ Most modes (CBC, CTR, GCM, ...) have a security proof like:

$$\text{Adv}_{\text{CBC-E}}^{\text{CPA}}(q, t) \leq \text{Adv}_E^{\text{PRP}}(q', t') + \frac{\sigma^2}{2^n}$$

- ▶ The CPA security of CBC is essentially the PRP security of E (the block cipher)
- ▶ As long as the **number of encrypted blocks** $\sigma \lll 2^{n/2}$
 - ▶ Usually matching attack with birthday complexity ($2^{n/2}$)
 - ▶ With a 64-bit cipher, the bound is **only 32 GB**

Communication issues

What cryptographers say

[Rogaway 2011]

"[birthday] attacks can be a serious concern when employing a blockcipher of $n = 64$ bits, requiring relatively frequent rekeying to keep $\sigma \ll 2^{32}$ "

What standards say

[ISO SC27 SD12]

"the maximum amount of plaintext that can be encrypted before rekeying must take place is $2^{(n/2)}$ blocks, due to the birthday paradox. As long as the implementation of a specific block cipher do not exceed these limits, using the block cipher will be safe."

What implementation do

TLS libraries, web browsers no rekeying

OpenVPN no rekeying (PSK mode) / rekey every hour (TLS mode)

Communication issues

What cryptographers say

[Rogaway 2011]

"[birthday] attacks can be a serious concern when employing a blockcipher of $n = 64$ bits, requiring relatively frequent rekeying to keep $\sigma \ll 2^{32}$ "

What standards say

[ISO SC27 SD12]

"the maximum amount of plaintext that can be encrypted before rekeying must take place is $2^{(n/2)}$ blocks, due to the birthday paradox. As long as the implementation of a specific block cipher do not exceed these limits, using the block cipher will be safe."

What implementation do

TLS libraries, web browsers no rekeying

OpenVPN no rekeying (PSK mode) / rekey every hour (TLS mode)

Communication issues

What cryptographers say

[Rogaway 2011]

"[birthday] attacks can be a serious concern when employing a blockcipher of $n = 64$ bits, requiring relatively frequent rekeying to keep $\sigma \ll 2^{32}$ "

What standards say

[ISO SC27 SD12]

"the maximum amount of plaintext that can be encrypted before rekeying must take place is $2^{(n/2)}$ blocks, due to the birthday paradox. As long as the implementation of a specific block cipher do not exceed these limits, using the block cipher will be safe."

What implementation do

TLS libraries, web browsers no rekeying

OpenVPN no rekeying (PSK mode) / rekey every hour (TLS mode)

Outline

Introduction

Towards a Practical attack

Attack against TLS

Impact and Mitigation

Impact

- ▶ How bad is it?
 - ▶ Is it bad to leak a few xors of blocks of plaintexts?
 - ▶ Do applications encrypt enough data under the same key?
- ▶ 64-bit block cipher are used in important protocols
 - ▶ With a 64-bit clock cipher, first collision around 32GB!
 - ▶ Blowfish-CBC in OpenVPN (default cipher)
 - ▶ 3DES-CBC in TLS (around 1-2%)
 - ▶ Kasumi in 3G (UMTS)
 - ▶ 64-bit ciphers with CBC were the norm before AES
- ▶ Collision attacks usually not considered a practical threat
 - ▶ openssl ciphers HIGH used to be sorted by key length
 - ▶ Before 2014: AES256, CAMELLIA256, 3DES, AES128, CAMELLIA128
 - ▶ After 2014: AES256, CAMELLIA256, AES128, CAMELLIA128, 3DES

Impact

- ▶ How bad is it?
 - ▶ Is it bad to leak a few xors of blocks of plaintexts?
 - ▶ Do applications encrypt enough data under the same key?
- ▶ 64-bit block cipher are used in important protocols
 - ▶ With a 64-bit clock cipher, first collision around 32GB!
 - ▶ Blowfish-CBC in **OpenVPN** (default cipher)
 - ▶ 3DES-CBC in **TLS** (around 1-2%)
 - ▶ Kasumi in **3G** (UMTS)
 - ▶ 64-bit ciphers with CBC were the norm before AES
- ▶ Collision attacks usually not considered a practical threat
 - ▶ openssl ciphers HIGH used to be sorted by key length
 - ▶ Before 2014: AES256, CAMELLIA256, **3DES**, AES128, CAMELLIA128
 - ▶ After 2014: AES256, CAMELLIA256, AES128, CAMELLIA128, **3DES**

Impact

Protocol	RFC	Year	Block ciphers	Mandatory	Rekey
TLS 1.0	2246	1999	3DES, DES, IDEA	3DES	-
TLS 1.1	4346	2006	AES, 3DES, DES	3DES	2^{78}
TLS 1.2	5246	2008	AES, 3DES	AES	2^{78}
SSH 1	draft	1995	3DES, DES, IDEA	3DES	-
SSH 2	4253	2006	AES, 3DES, Blowfish	3DES	2^{30}
IKEv1	2409	1998	3DES, DES, Blowfish	DES	-
IKEv2	6996	2010	AES, 3DES, Blowfish	3DES	-
IPsec	7321	2014	AES, 3DES	AES	1GB

Impact

- ▶ How bad is it?
 - ▶ Is it bad to leak a few xors of blocks of plaintexts?
 - ▶ Do applications encrypt enough data under the same key?
- ▶ 64-bit block cipher are used in important protocols
 - ▶ With a 64-bit clock cipher, first collision around 32GB!
 - ▶ Blowfish-CBC in **OpenVPN** (default cipher)
 - ▶ 3DES-CBC in **TLS** (around 1-2%)
 - ▶ Kasumi in **3G** (UMTS)
 - ▶ 64-bit ciphers with CBC were the norm before AES
- ▶ Collision attacks usually not considered a practical threat
 - ▶ openssl ciphers HIGH used to be sorted by key length
 - ▶ Before 2014: AES256, CAMELLIA256, **3DES**, AES128, CAMELLIA128
 - ▶ After 2014: AES256, CAMELLIA256, AES128, CAMELLIA128, **3DES**

Towards a practical attack

- ▶ Assume a **fixed message** is **repeatedly** encrypted (under a **fixed key**)
 - ▶ Including a high value secret (cookie, password, ...) a few blocks
 - ▶ And some known/predictable sections (headers, ...) 2^t blocks
- ▶ Each collision reveals the xor of two plaintext blocks
- ▶ Eventually a collision will reveal the secret
- ▶ Success after roughly 2^t collisions
 - ▶ $2^{n/2-t/2}$ message copies, $2^{n/2+t/2}$ blocks
 - ▶ Tradeoff between number of copies and total amount of data
- ▶ If rekeying after roughly $2^{n/2}$ blocks, attack still possible
 - ▶ $2^{n/2}$ message copies, $2^{n/2+t}$ blocks

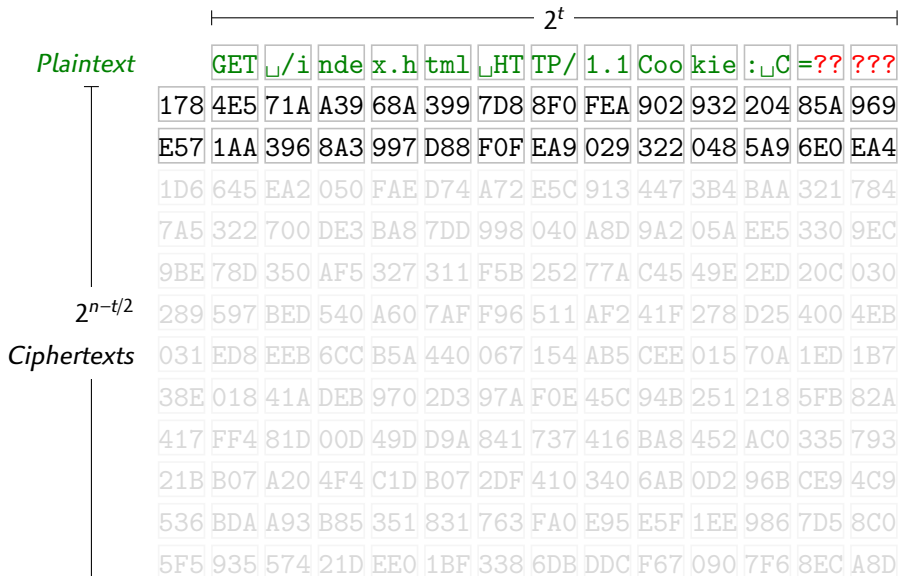
Towards a practical attack



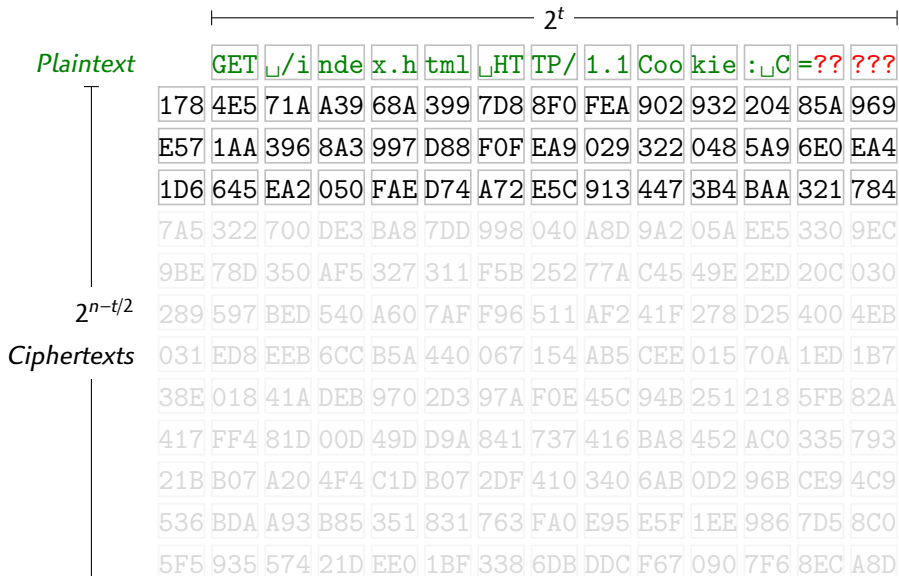
Towards a practical attack

		2^t												
<i>Plaintext</i>		GET	_/i	nde	x.h	tml	_HT	TP/	1.1	Coo	kie	:_C	=??	???
<i>Ciphertexts</i>	178	4E5	71A	A39	68A	399	7D8	8F0	FEA	902	932	204	85A	969
	E57	1AA	396	8A3	997	D88	F0F	EA9	029	322	048	5A9	6E0	EA4
	1D6	645	EA2	050	FAE	D74	A72	E5C	913	447	3B4	BAA	321	784
	7A5	322	700	DE3	BA8	7DD	998	040	A8D	9A2	05A	EE5	330	9EC
	9BE	78D	350	AF5	327	311	F5B	252	77A	C45	49E	2ED	20C	030
	289	597	BED	540	A60	7AF	F96	511	AF2	41F	278	D25	400	4EB
	031	ED8	EEB	6CC	B5A	440	067	154	AB5	CEE	015	70A	1ED	1B7
	38E	018	41A	DEB	970	2D3	97A	FOE	45C	94B	251	218	5FB	82A
	417	FF4	81D	00D	49D	D9A	841	737	416	BA8	452	AC0	335	793
	21B	B07	A20	4F4	C1D	B07	2DF	410	340	6AB	0D2	96B	CE9	4C9
	536	BDA	A93	B85	351	831	763	FA0	E95	E5F	1EE	986	7D5	8C0
	5F5	935	574	21D	EE0	1BF	338	6DB	DDC	F67	090	7F6	8EC	A8D

Towards a practical attack



Towards a practical attack



Towards a practical attack

Plaintext

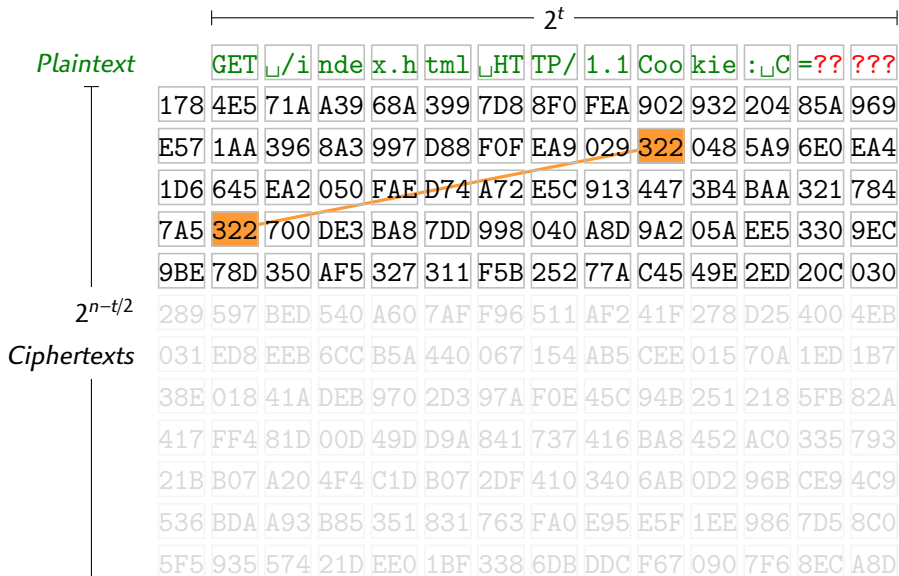
2^t

$2^{n-t/2}$

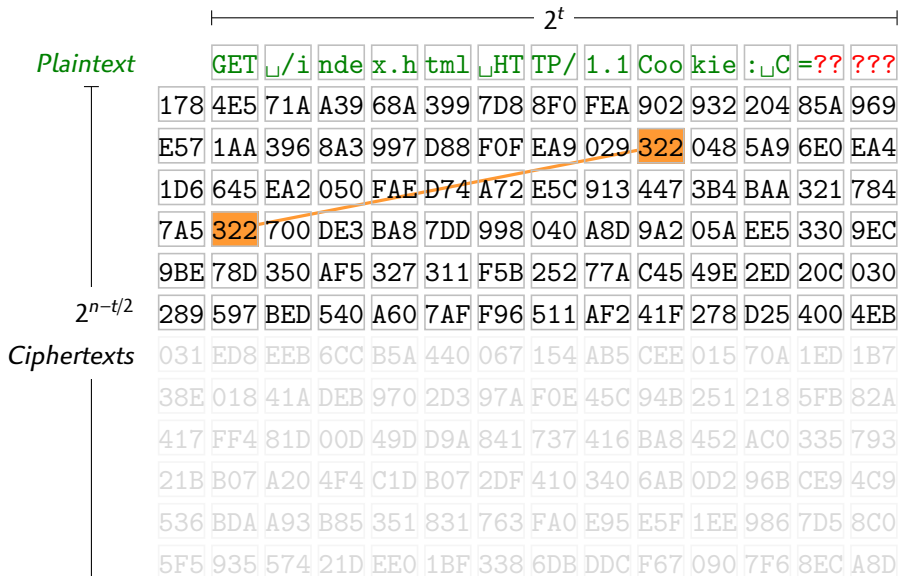
Ciphertexts

	GET	/i	nde	x.h	tml	/HT	TP/	1.1	Coo	kie	:_C	=??	???
178	4E5	71A	A39	68A	399	7D8	8F0	FEA	902	932	204	85A	969
E57	1AA	396	8A3	997	D88	F0F	EA9	029	322	048	5A9	6E0	EA4
1D6	645	EA2	050	FAE	D74	A72	E5C	913	447	3B4	BAA	321	784
7A5	322	700	DE3	BA8	7DD	998	040	A8D	9A2	05A	EE5	330	9EC
9BE	78D	350	AF5	327	311	F5B	252	77A	C45	49E	2ED	20C	030
289	597	BED	540	A60	7AF	F96	511	AF2	41F	278	D25	400	4EB
031	ED8	EEB	6CC	B5A	440	067	154	AB5	CEE	015	70A	1ED	1B7
38E	018	41A	DEB	970	2D3	97A	F0E	45C	94B	251	218	5FB	82A
417	FF4	81D	00D	49D	D9A	841	737	416	BA8	452	AC0	335	793
21B	B07	A20	4F4	C1D	B07	2DF	410	340	6AB	0D2	96B	CE9	4C9
536	BDA	A93	B85	351	831	763	FA0	E95	E5F	1EE	986	7D5	8C0
5F5	935	574	21D	EE0	1BF	338	6DB	DDC	F67	090	7F6	8EC	A8D

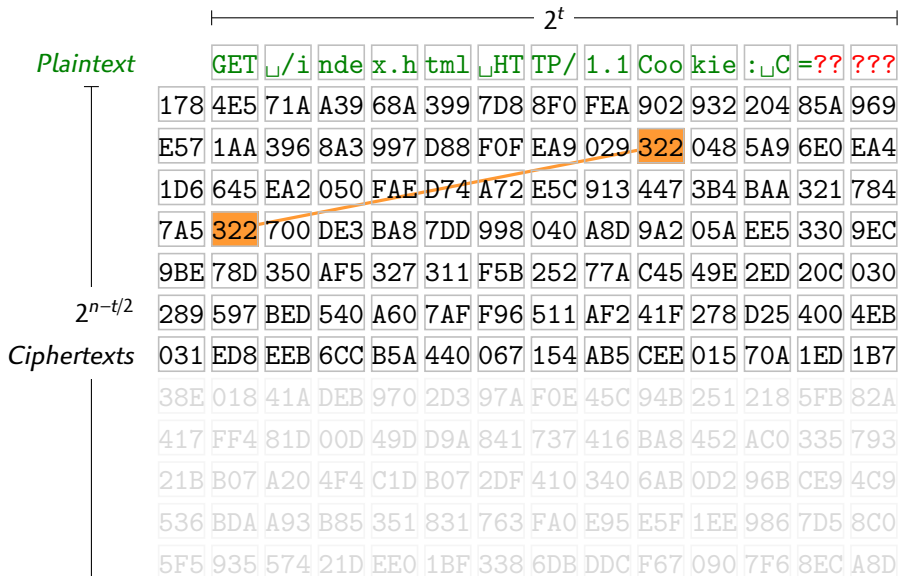
Towards a practical attack



Towards a practical attack



Towards a practical attack



Towards a practical attack

2^t

Plaintext

GET /index.html HTTP/1.1 Cookie: C=?? ???

178	4E5	71A	A39	68A	399	7D8	8F0	FEA	902	932	204	85A	969	
E57	1AA	396	8A3	997	D88	F0F	EA9	029	322	048	5A9	6E0	EA4	
1D6	645	EA2	050	FAE	D74	A72	E5C	913	447	3B4	BAA	321	784	
7A5	322	700	DE3	BA8	7DD	998	040	A8D	9A2	05A	EE5	330	9EC	
9BE	78D	350	AF5	327	311	F5B	252	77A	C45	49E	2ED	20C	030	
$2^{n-t/2}$	289	597	BED	540	A60	7AF	F96	511	AF2	41F	278	D25	400	4EB
<i>Ciphertexts</i>	031	ED8	EEB	6CC	B5A	440	067	154	AB5	CEE	015	70A	1ED	1B7
	38E	018	41A	DEB	970	2D3	97A	FOE	45C	94B	251	218	5FB	82A
	417	FF4	81D	00D	49D	D9A	841	737	416	BA8	452	AC0	335	793
	21B	B07	A20	4F4	C1D	B07	2DF	410	340	6AB	0D2	96B	CE9	4C9
	536	BDA	A93	B85	351	831	763	FA0	E95	E5F	1EE	986	7D5	8C0
	5F5	935	574	21D	EE0	1BF	338	6DB	DDC	F67	090	7F6	8EC	A8D

Towards a practical attack

2^t

Plaintext

GET /index.html HTTP/1.1 Cookie: C=?? ???

178	4E5	71A	A39	68A	399	7D8	8F0	FEA	902	932	204	85A	969	
E57	1AA	396	8A3	997	D88	F0F	EA9	029	322	048	5A9	6E0	EA4	
1D6	645	EA2	050	FAE	D74	A72	E5C	913	447	3B4	BAA	321	784	
7A5	322	700	DE3	BA8	7DD	998	040	A8D	9A2	05A	EE5	330	9EC	
9BE	78D	350	AF5	327	311	F5B	252	77A	C45	49E	2ED	20C	030	
$2^{n-t/2}$	289	597	BED	540	A60	7AF	F96	511	AF2	41F	278	D25	400	4EB
Ciphertexts	031	ED8	EEB	6CC	B5A	440	067	154	AB5	CEE	015	70A	1ED	1B7
	38E	018	41A	DEB	970	2D3	97A	F0E	45C	94B	251	218	5FB	82A
	417	FF4	81D	00D	49D	D9A	841	737	416	BA8	452	AC0	335	793
	21B	B07	A20	4F4	C1D	B07	2DF	410	340	6AB	0D2	96B	CE9	4C9
	536	BDA	A93	B85	351	831	763	FA0	E95	E5F	1EE	986	7D5	8C0
	5F5	935	574	21D	EE0	1BF	338	6DB	DDC	F67	090	7F6	8EC	A8D

Towards a practical attack

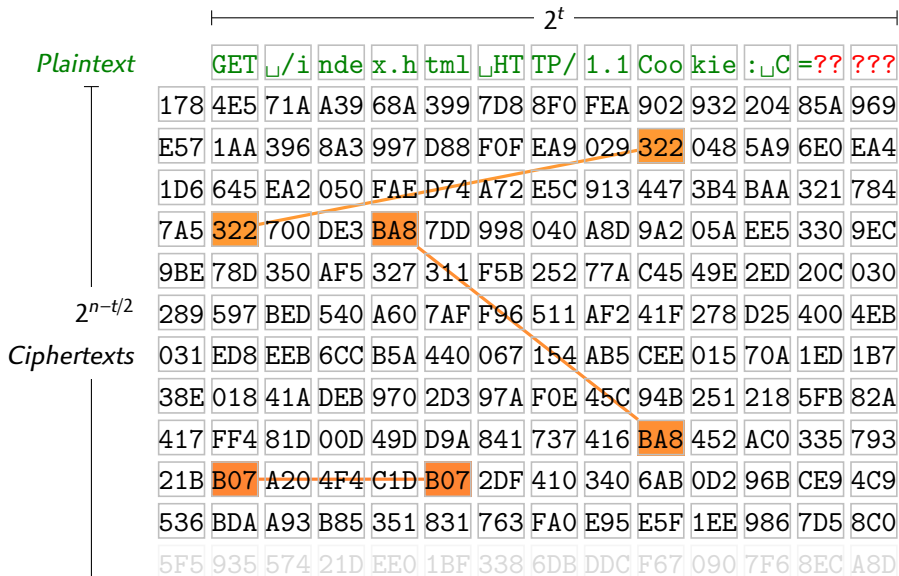
2^t

Plaintext

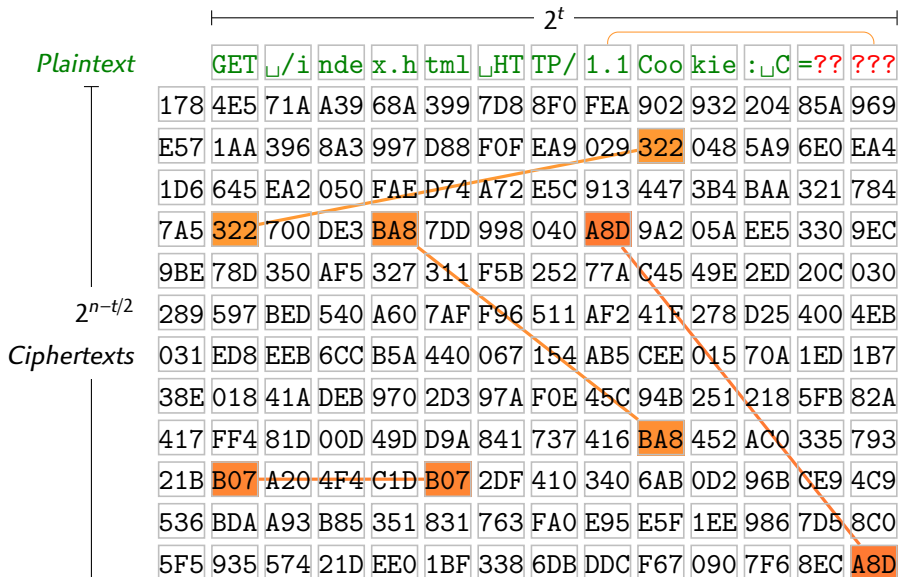
GET /index.html HTTP/1.1 Cookie: C=?? ???

178	4E5	71A	A39	68A	399	7D8	8F0	FEA	902	932	204	85A	969	
E57	1AA	396	8A3	997	D88	F0F	EA9	029	322	048	5A9	6E0	EA4	
1D6	645	EA2	050	FAE	D74	A72	E5C	913	447	3B4	BAA	321	784	
7A5	322	700	DE3	BA8	7DD	998	040	A8D	9A2	05A	EE5	330	9EC	
9BE	78D	350	AF5	327	311	F5B	252	77A	C45	49E	2ED	20C	030	
$2^{n-t/2}$	289	597	BED	540	A60	7AF	F96	511	AF2	41F	278	D25	400	4EB
<i>Ciphertexts</i>	031	ED8	EEB	6CC	B5A	440	067	154	AB5	CEE	015	70A	1ED	1B7
	38E	018	41A	DEB	970	2D3	97A	F0E	45C	94B	251	218	5FB	82A
	417	FF4	81D	00D	49D	D9A	841	737	416	BA8	452	AC0	335	793
	21B	B07	A20	4F4	C1D	B07	2DF	410	340	6AB	0D2	96B	CE9	4C9
	536	BDA	A93	B85	351	831	763	FA0	E95	E5F	1EE	986	7D5	8C0
	5F5	935	574	21D	EE0	1BF	338	6DB	DDC	F67	090	7F6	8EC	A8D

Towards a practical attack



Towards a practical attack



Towards a practical attack

- ▶ Assume a **fixed message** is **repeatedly** encrypted (under a **fixed key**)
 - ▶ Including a high value secret (cookie, password, ...) a few blocks
 - ▶ And some known/predictable sections (headers, ...) 2^t blocks
- ▶ Each collision reveals the xor of two plaintext blocks
- ▶ Eventually a collision will reveal the secret
- ▶ Success after roughly 2^t collisions
 - ▶ $2^{n/2-t/2}$ **message copies**, $2^{n/2+t/2}$ **blocks**
 - ▶ Tradeoff between number of copies and total amount of data
- ▶ If rekeying after roughly $2^{n/2}$ blocks, attack still possible
 - ▶ $2^{n/2}$ message copies, $2^{n/2+t}$ blocks

Towards a practical attack

- ▶ Assume a **fixed message** is **repeatedly** encrypted (under a **fixed key**)
 - ▶ Including a high value secret (cookie, password, ...) a few blocks
 - ▶ And some known/predictable sections (headers, ...) 2^t blocks
- ▶ Each collision reveals the xor of two plaintext blocks
- ▶ Eventually a collision will reveal the secret
- ▶ Success after roughly 2^t collisions
 - ▶ 2^{n/2-t/2} **message copies**, 2^{n/2+t/2} **blocks**
 - ▶ Tradeoff between number of copies and total amount of data
- ▶ If rekeying after roughly 2^{n/2} blocks, attack still possible
 - ▶ 2^{n/2} message copies, 2^{n/2+t} blocks

HTTP authentication tokens

- ▶ HTTP is stateless: authentication tokens sent **with every request**
- ▶ Also sent with *cross-origin* requests to allow “Facebook button”

HTTP Basic Auth (RFC 7617)

- ▶ User/Password sent in a header (base64 encoded)

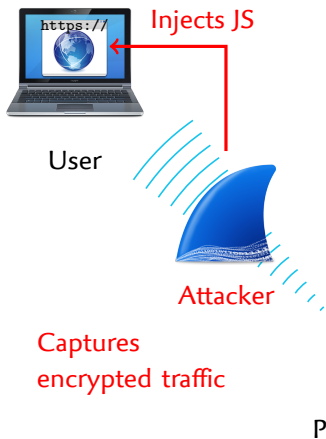
Authorization: Basic dGVzdDoxMjPCow=

HTTP Cookies (RFC 6265)

- 1 User sends password in a from
- 2 Server reply with a Cookie
- 3 Cookie is included in every subsequent request

Cookie: C=123456

Beastly Attack Scenario

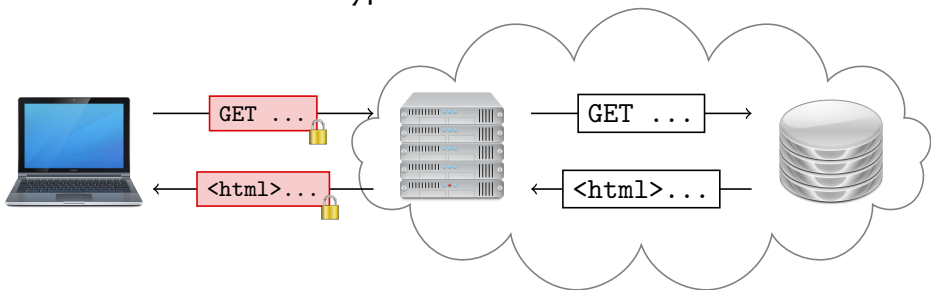


- ▶ Attacker has access to the network (e.g. public WiFi)
 - ▶ User logged-in to secure website (w/ cookie or BasicAuth)
- 1 Attacker uses JS to generate traffic
 - ▶ Tricks victim to malicious site
 - ▶ JS makes *cross-origin* requests
 - 2 Attacker captures encrypted data

[BEAST, Duong & Rizzo 2011]

OpenVPN

- ▶ A VPN creates an encrypted tunnel to between two machines



- ▶ **OpenVPN** is a popular free-software VPN solution
 - ▶ Default cipher: Blowfish in CBC mode (64-bit blocks)
 - ▶ *Pre-shared-key mode*: no rekeying
 - ▶ *TLS mode*: rekeying every hour (by default), 2^{32} packets limit

Proof-of-concept Attack Demo: HTTP over OpenVPN

- ▶ Demo with **Firefox** browser (Linux), and **nginx** server connected with **OpenVPN** in pre-shared-key mode
 - ▶ Default configuration
- ▶ Each HTTP request encrypted in OpenVPN packet, with fixed key

- 1** Generate traffic with malicious JavaScript
 - ▶ Use 4kB requests (pad URL or cookie)
 - ▶ About 2900 requests/second
 - 2** Capture on the network with `tcpdump`
 - 3** Remove header, extract ciphertext at fixed position
 - 4** Sort ciphertext (`std::xx1`), look for collisions
- ▶ **Expected time:** 19 hours for 785 GB.
 - ▶ **In practice:** 18.6 hours for 705 GB.

Outline

Introduction

Towards a Practical attack

Attack against TLS

Impact and Mitigation

HTTPS: HTTP over TLS

- ▶ **HTTPS: secure HTTP**
 - ▶ HTTP over a TLS connection
 - ▶ One of the most widespread use of encryption
- ▶ TLS is **agile**: ciphersuite negotiation
 - ▶ Client sends ordered list of supported ciphersuites
 - ▶ Server chooses ciphersuite
 - ▶ Most servers force their ordering

 - ▶ Block cipher key derived from key exchange
- ▶ **3DES** is one of the possible ciphers
 - ▶ Mandatory to implement up to TLS 1.1
 - ▶ How much is used?
 - ▶ How much data can be encrypted with the same key?

3DES use in TLS (HTTPS)

- ▶ It seems that **1–2%** of HTTPS connections **use 3DES**
 - ▶ Outdated client/servers
 - ▶ Windows XP / Windows 2003 Server don't support AES out of the box
 - ▶ Many **poorly configured servers** support AES, but prefer 3DES
- ▶ Scan of Alexa's top 1 million websites
 - ▶ 3DES use assuming a modern browser (AES > 3DES, no RC4)

3DES	February 2016		October 2016	
	support	use	support	use
Top 1k	93%	1.6%	84%	1.5%
Top 10k	92%	2.1%	84%	1.0%
Top 100k	89%	1.9%	83.7%	0.9%
Top 1M	86%	1.3%	86%	1.0%

Poorly configured websites

ebay.com

The screenshot shows a Mozilla Firefox browser window displaying the eBay sign-in page. The address bar shows the URL `https://signin.ebay.com/ws/`. The page features the eBay logo and a sign-in form with fields for email/username and password, and a 'Sign in' button. A 'Page Info' window is open over the sign-in form, showing details about the connection to `signin.ebay.com`. The 'Web Site Identity' section lists the owner as eBay, Inc. and the verifier as Symantec Corporation. The 'Privacy & History' section shows that the user has visited the site 3 times and that the site stores cookies and saved passwords. The 'Technical Details' section indicates that the connection is encrypted using TLS 1.2 with 128-bit keys. A red stamp 'Fixed in October 2016' is overlaid on the Page Info window, and a red circle highlights the text 'before being transmitted' in the 'Technical Details' section.

Poorly configured websites

match.com

Match.com® | Login | The Leading Online Dating Site for Singles & Personals : Match.com - Mozilla Firefox

File Edit View History Bookmarks Tools Help

Match.com® | Login | ... x +

MATCH.COM, L.L.C. (US) | https://www4.match.com/login/

match SUBSCRIBE

Sign in to continue...

enter email

enter password

SIGN IN NOW

Keep me signed in.
Don't check it if you're on a public/shared computer.

New to Match.com? Join

About Match.com | Online Dating S

Page Info - https://www4.match.com/login/

General Media Feeds Permissions Security

Web Site Identity

Web site: **www4.match.com**
 Owner: **MATCH.COM, L.L.C.**
 Verified by: **Symantec Corporation**

Privacy & History

Have I visited this web site before today?	No
Is this web site storing information (cookies) on my computer?	Yes
Have I saved any passwords for this web site?	No

Technical Details

Connection Encrypted (TLS_RSA_WITH_3DES_EDE_CBC_SHA, 112 bit keys, TLS 1.2)
 The page you are viewing was encrypted before being transmitted over the internet.
 Encryption makes it difficult for unauthorised people to view information travelling between computers. It is therefore unlikely that anyone read this page as it travelled across the network.

View Certificate

View Cookies

View Saved Passwords

Help

Poorly configured websites

match.com

<https://discovery.cryptosense.com/analyze/208.83.241.15>



208.83.241.15

IP address **208.83.241.15**

Last scan **2016-10-20 12:29:18 UTC**

TLS HTTP (port 443)

Rules applicable **13**

B	A	A'	B	C	D
9	2	2	0	0	0

TLS (port 443 – HTTP)

Show scan details ▾

Versions **TLS 1.0, TLS 1.1**

Fallback SCSV **Not supported**

Ciphers

TLS_RSA_WITH_3DES_EDE_CBC_SHA	TLS 1.0, TLS 1.1
TLS_RSA_WITH_AES_128_CBC_SHA	TLS 1.0, TLS 1.1
TLS_RSA_WITH_AES_256_CBC_SHA	TLS 1.0, TLS 1.1

Poorly configured websites

webmail.trumporg.com

<https://discovery.cryptosense.com/analyze/trumporg.com>



webmail.trumporg.com

IP address 192.154.117.35

Last scan 2016-10-20 12:07:27 UTC

TLS HTTP (port 443)

Rules applicable 12

D	A	A'	B	C	D
4	2	1	1	4	

TLS (port 443 – HTTP)

Show scan details ▾

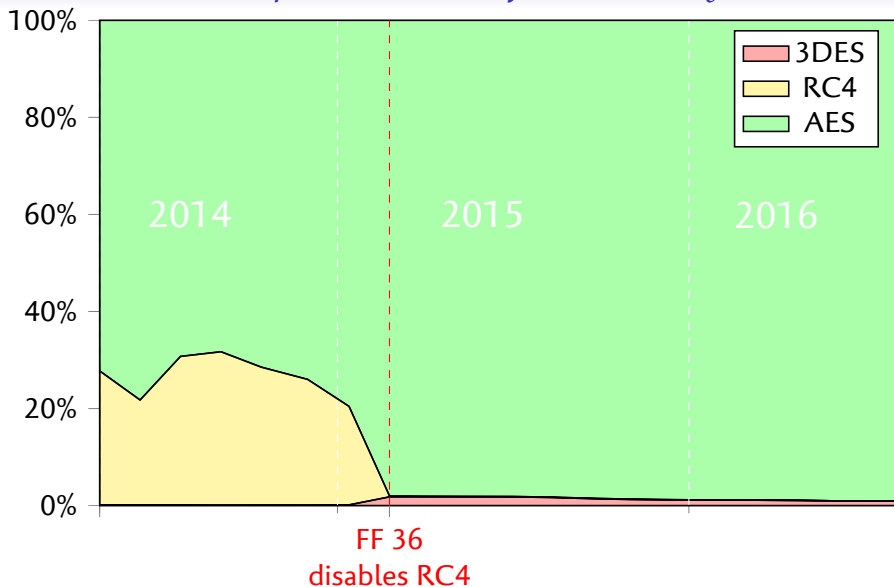
Versions

SSL 2.0, TLS 1.0

Ciphers

TLS_RSA_WITH_RC4_128_MD5 TLS 1.0
 TLS_RSA_WITH_RC4_128_SHA TLS 1.0
 TLS_RSA_WITH_3DES_EDE_CBC_SHA TLS 1.0
 TLS_RSA_WITH_DES_CBC_SHA TLS 1.0
 TLS_RSA_EXPORT1024_WITH_RC4_56_SHA TLS 1.0
 TLS_RSA_EXPORT1024_WITH_DES_CBC_SHA TLS 1.0
 TLS_RSA_EXPORT_WITH_RC4_40_MD5 TLS 1.0
 TLS_RSA_EXPORT_WITH_RC2_CBC_40_MD5 TLS 1.0
 SSL2_RC4_128_WITH_MD5 SSL 2.0
 SSL2_DES_192_EDE3_CBC_WITH_MD5 SSL 2.0
 SSL2_RC2_128_CBC_WITH_MD5 SSL 2.0
 SSL2_DES_64_CBC_WITH_MD5 SSL 2.0
 SSL2_RC4_128_EXPORT40_WITH_MD5 SSL 2.0
 SSL2_RC4_128_CBC_EXPORT40_WITH_MD5 SSL 2.0

TLS cipher use in Firefox (telemetry)



HTTPS session length

- ▶ HTTP 1.0 uses one connection per request
- ▶ HTTP 1.1 can reuse a connection (Keep-alive)
- ▶ *Web browsers* reuse a connection as long as possible
- ▶ *Web servers*
 - Apache* has a limit on connection reuse (default 200)
 - Nginx* has a limit on connection reuse (default 200)
 - IIS* doesn't have limit by default
- ▶ In practice, many high-profile website support very long sessions
- ▶ We found many **vulnerable websites** that
 - ▶ Use 3DES with a modern browser
 - ▶ Support very long sessions (> 1M)

HTTPS session length

- ▶ HTTP 1.0 uses one connection per request
- ▶ HTTP 1.1 can reuse a connection (Keep-alive)
- ▶ *Web browsers* reuse a connection as long as possible
- ▶ *Web servers*
 - Apache* has a limit on connection reuse (default 200)
 - Nginx* has a limit on connection reuse (default 200)
 - IIS* doesn't have limit by default
- ▶ In practice, many high-profile website support very long sessions
- ▶ We found many **vulnerable websites** that
 - ▶ Use 3DES with a modern browser
 - ▶ Support very long sessions (> 1M)

Proof-of-concept Attack Demo: HTTPS

- ▶ Demo with **Firefox** (Linux), and **IIS 6.0** (Windows Server 2003)
 - ▶ Default configuration of IIS 6.0 does not support AES
- ▶ Each HTTP request encrypted in TLS record, with fixed key

1 Generate traffic with malicious JavaScript

- ▶ Use 4kB requests (pad URL or cookie)
- ▶ About 1500 requests/second

2 Capture on the network with tcpdump

- ▶ Tamper with traffic to have a single active connection

3 Remove header, extract ciphertext at fixed position

4 Sort ciphertext (stdxx1), look for collisions

- ▶ **Expected time:** 38 hours for 785 GB.
- ▶ **In practice:** 30.5 hours for 610 GB.

Outline

Introduction

Towards a Practical attack

Attack against TLS

Impact and Mitigation

Countermeasures

- ▶ Switch to **128-bit block ciphers** (e.g. AES)
 - ▶ Fix server TLS config
- ▶ Limit connection length
 - ▶ Can be done on client or server independently
- ▶ Use a beyond-birthday-secure mode (e.g. CENC)
 - ▶ Could be an option for lightweight crypto

Should we get rid of 3DES in TLS?

- ▶ Make sure it's only used as a **last resort**, and **use rekeying**
- ▶ Even then, having it available is a **potential weakness**
 - ▶ There might be downgrade attacks
 - ▶ Example: 3DES can be forced if TLS false start

Countermeasures

- ▶ Switch to **128-bit block ciphers** (e.g. AES)
 - ▶ Fix server TLS config
- ▶ Limit connection length
 - ▶ Can be done on client or server independently
- ▶ Use a beyond-birthday-secure mode (e.g. CENC)
 - ▶ Could be an option for lightweight crypto

Should we get rid of 3DES in TLS?

- ▶ Make sure it's only used as a **last resort**, and **use rekeying**
- ▶ Even then, having it available is a **potential weakness**
 - ▶ There might be downgrade attacks
 - ▶ Example: 3DES can be forced if TLS false start

Disclosure

Sweet32 attack disclosed on August 24

- ▶ <https://sweet32.info>
- ▶ CVE-2016-2183, CVE-2016-6329



- ▶ **OpenVPN** 2.3.12 issues a warning when using 64-bit block cipher
 - ▶ Future versions will implement connection limit, and cipher negotiation defaulting to AES
- ▶ **Mozilla** has implemented data limits in NSS 3.27 (1M records)
- ▶ **OpenSSL** moved 3DES to LOW category
- ▶ **Microsoft** removed 3DES from False Start white-list
- ▶ Some websites fixed their TLS configuration

Comparison with RC4 attacks

Practical attacks against TLS with RC4

[AFBPPS, Usenix '13]

- ▶ With a **different key each session**
 - ▶ Using biases in the RC4 keystream
 - ▶ Plaintext recovery (220 first bytes) with $2^{28} - 2^{32}$ sessions
- ▶ With longer sessions
 - ▶ Using Fluhrer-McGrew biases (single or multiple sessions)
 - ▶ Cookie recovery with $2^{33} - 2^{34}$ requests
 - ▶ Latest improvement: $2^{30.2}$ requests [Vanhoef & Piessens, Usenix '15]

Practical attack against TLS with 3DES

- ▶ Using a single **long-lived session**
- ▶ $2^{29.1}$ short query (512 bytes) 280 GB total
- ▶ Or $2^{27.6}$ longer queries (4 kB) 785 GB total

Conclusion

Block size does matter

- ▶ **Birthday attack** against CBC with $2^{n/2}$ data
- ▶ Attacks with 2^{32} data are **practical**
- ▶ Independent of key size, block cipher strength
 - ▶ 64-bit block ciphers (3DES, Blowfish)
not much more secure than RC4



- ▶ Protocols designed in the 90's still use 64-bit ciphers
- ▶ Demo of two practical attacks
 - ▶ Blowfish default cipher in OpenVPN
 - ▶ Badly configured HTTPS servers use 3DES

<https://sweet32.info>