How Not to Use a Blockcipher

Gaëtan Leurent

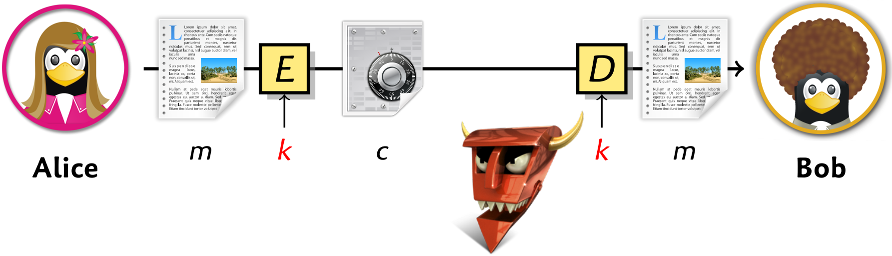
Inria

COST Training School, Feb. 2018

Block ciphers

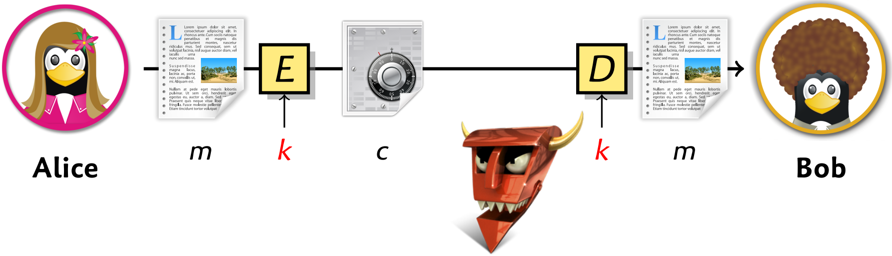
- Alice and Bob want to communicate securely
  - Confidentiality
  - Integrity
- They've heard about block ciphers... How to use them?

Block ciphers



- Alice and Bob want to communicate securely
  - Confidentiality
  - Integrity
- They've heard about block ciphers... How to use them?

Block ciphers



- Alice and Bob want to communicate securely
  - Confidentiality
  - Integrity
- They've heard about block ciphers... How to use them?

Block ciphers

- A block cipher is a family of permutations
  - Should behave like a set of 2^κ random permutations (out of $(2^n)!$)
- Great if Alice has a single message of n bits
  - How to deal with a message longer than n-bits?
  - How to deal with several messages?

Naive solution: Electronic Code Book (ECB)

- Divide message into n-bit blocks: $M = m_1 \| m_2 \| \ldots$
- Encrypt block independently: $C = E(m_1) \| E(m_2) \| \ldots$

Block ciphers

- A block cipher is a family of permutations
  - Should behave like a set of 2^κ random permutations (out of $(2^n)!$)
- Great if Alice has a single message of n bits
  - How to deal with a message longer than n-bits?
  - How to deal with several messages?

Naive solution: Electronic Code Book (ECB)

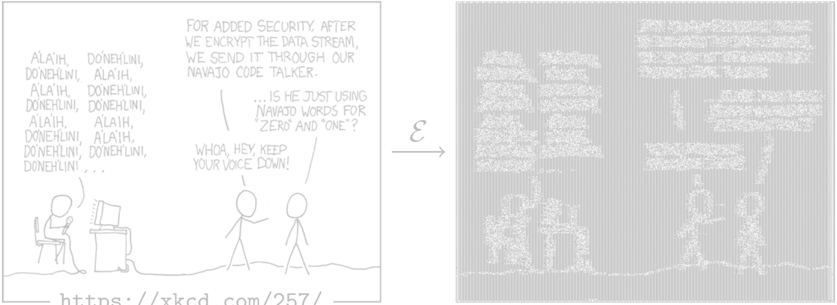
- Divide message into n-bit blocks: $M = m_1 \| m_2 \| \ldots$
- Encrypt block independently: $C = E(m_1) \| E(m_2) \| \ldots$

- Problem: If two blocks are equal, the encryption is the same

$$m_i = m_j \implies E(m_i) = E(m_j)$$

ECB issues

- Formatted messages often have low entropy
  - Bitmap images
  - HTML text with tags
  - Headers
  - …



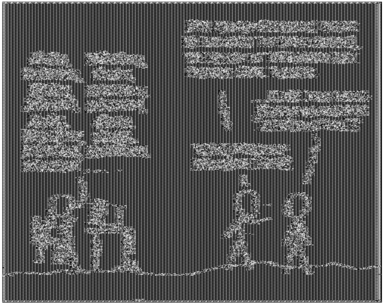
https://xkcd.com/257/

ECB issues

- Formatted messages often have low entropy
  - Bitmap images
  - HTML text with tags
  - Headers
  - …

https://xkcd.com/257/

$\xrightarrow{\mathcal{E}}$



Security Notions

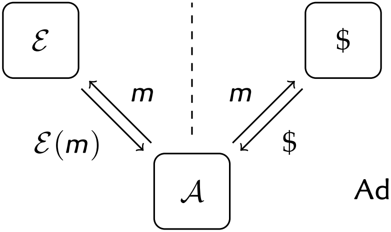
- We aim for computational security
  - Perfect security requires a key as large as the message
- Attack should be impossible in practice
- Security goal: No attack with less than X operations, with large X
  - X defined with generic attacks: *e.g.* exhaustive key search

Orders of magnitude

- Largest cryptanalitic attack: SHA-1 collision — 2^{63} SHA-1
- Bitcoin network — 2^{74} SHA-256 / hr
- Google storage — $\approx 2^{64}$ bytes

Meet the adversary

- Attacker has access to some information
  - Ciphertext only
  - Ciphertext with Known plaintext
  - Ciphertext with Chosen plaintext (encryption oracle)
- Attacker must break some security notion
  - Key recovery
  - Plaintext recovery
  - Distinguish ciphertext from random
- Focus on strongest notion: distinguisher with chosen plaintext:



$$\mathrm{Adv}(\mathcal{A}) = \left|\Pr[\mathcal{A}^{\mathcal{E}} \to 1] - \Pr[\mathcal{A}^{\$} \to 1]\right|$$

HTTPS encryption: HTTP over TLS

HTTP

- Hypertext Transfer Protocol
  - Request/response (text)
  - Headers and body

```
GET /index.html HTTP/1.1
User-Agent: Firefox
```

```
HTTP/1.1 200 OK
Content-Type: text/html

<html>
  <body>...
```

TLS

- Transport Layer Security
  - Evolution of Netscape's SSL
  - Current version: TLS 1.2
- Stream encryption protocol
  - Algorithm negotiation
  - Handshake: asym. crypto
  - Transport: sym. crypto
- Each HTTP message encrypted in a TLS packet

HTTP authentication tokens

- HTTP is stateless: authentication tokens sent with every request
  - HTTP 1.1 Keep-alive sends many requests in the same connection

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
```

Javascript attack

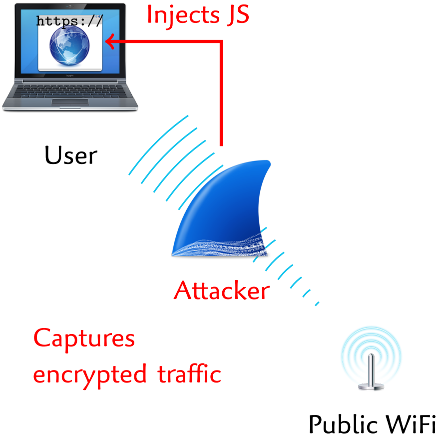
- A webpage is not just data, it includes code
- Malicious website can send requests to third party
- Requests include authentication cookies

Javascipt attack

```
var url = "https://www.facebook.com/index.html";
var xhr = new XMLHttpRequest;

while(true) {
    xhr.open("HEAD", url, false);
    xhr.withCredentials = true;
    xhr.send();
    xhr.abort();
}
```

BEAST Attack Setting [Duong & Rizzo 2011]



- Attacker has access to the network (*eg.* public WiFi)

1. Attacker uses JS to generate traffic
   - Tricks victim to malicious site
   - JS makes *cross-origin* requests
2. Attacker captures encrypted data

- Very powerful model
  Chosen plaintext

Chosen-Prefix Secret-Suffix [Hoang & *al.*, Crypto'15]

- We can model these attacks as Chosen-Prefix Secret-Suffix
  - Fixed secret high-value S
  - Oracle access $M \mapsto \mathcal{E}(M \| S)$
    - Secret included in the message

Exercise: Message recovery

Can we recover S in this model with ECB encryption?

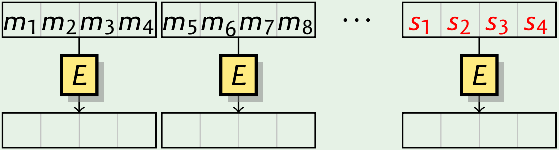
Chosen-Prefix Secret-Suffix

[Hoang & *al.*, Crypto'15]

- We can model these attacks as Chosen-Prefix Secret-Suffix
  - Fixed secret high-value S
  - Oracle access $M \mapsto \mathcal{E}(M \| S)$
    - Secret included in the message

Exercise: Message recovery

Can we recover S in this model with ECB encryption?

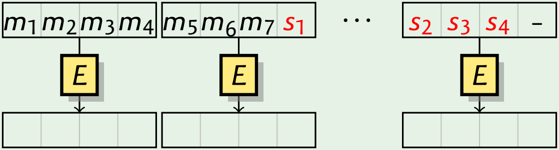


Chosen-Prefix Secret-Suffix [Hoang & *al.*, Crypto'15]

- We can model these attacks as Chosen-Prefix Secret-Suffix
  - Fixed secret high-value S
  - Oracle access $M \mapsto \mathcal{E}(M \| S)$
    - Secret included in the message

Exercise: Message recovery

Can we recover S in this model with ECB encryption?



Chosen-Prefix Secret-Suffix

[Hoang & *al.*, Crypto'15]

1. Use a prefix of length $\ell = n - 1 \bmod n$
   Guess last block: single unknown byte

0 0 0 0	0 0 0 1	0 0 0 2	0 0 0 3	$\cdots$	0 0 0 s_0	s_1 s_2 s_3 –

2. When guess is correct, collision reveals s_0
3. Use a prefix of length $\ell = n - 2 \bmod n$:

0 0 s_0 0	0 0 s_0 1	0 0 s_0 2	0 0 s_0 3	$\cdots$	0 0 s_0 s_1	s_2 s_3 – –

4. Collision reveals s_1
5. Iterate . . .

Chosen-Prefix Secret-Suffix [Hoang & *al.*, Crypto'15]

1. Use a prefix of length $\ell = n - 1 \bmod n$
   Guess last block: single unknown byte

   | 0 0 0 0 | 0 0 0 1 | 0 0 0 2 | 0 0 0 3 | ⋯ | 0 0 0 s_0 | s_1 s_2 s_3 – |

2. When guess is correct, collision reveals s_0
3. Use a prefix of length $\ell = n - 2 \bmod n$:

   | 0 0 s_0 0 | 0 0 s_0 1 | 0 0 s_0 2 | 0 0 s_0 3 | ⋯ | 0 0 s_0 s_1 | s_2 s_3 – – |

4. Collision reveals s_1
5. Iterate …

Chosen-Prefix Secret-Suffix

[Hoang & al., Crypto'15]

1. Use a prefix of length $\ell = n - 1 \bmod n$
   Guess last block: single unknown byte

0 0 0 0	0 0 0 1	0 0 0 2	0 0 0 3	$\cdots$	0 0 0 s_0	s_1 s_2 s_3 -

2. When guess is correct, collision reveals s_0
3. Use a prefix of length $\ell = n - 2 \bmod n$:

0 0 s_0 0	0 0 s_0 1	0 0 s_0 2	0 0 s_0 3	$\cdots$	0 0 s_0 s_1	s_2 s_3 - -

4. Collision reveals s_1
5. Iterate …

Chosen-Prefix Secret-Suffix [Hoang & *al.*, Crypto'15]

1. Use a prefix of length $\ell = n - 1 \bmod n$
   Guess last block: single unknown byte

0 0 0 0	0 0 0 1	0 0 0 2	0 0 0 3	$\cdots$	0 0 0 s_0	s_1 s_2 s_3 –

2. When guess is correct, collision reveals s_0
3. Use a prefix of length $\ell = n - 2 \bmod n$:

0 0 s_0 0	0 0 s_0 1	0 0 s_0 2	0 0 s_0 3	$\cdots$	0 0 s_0 s_1	s_2 s_3 – –

4. Collision reveals s_1
5. Iterate …

How Not to Use a Blockcipher

- Even with a secure block cipher, secure communication is not easy
  - Block cipher modes for encryption and authentication
- This lecture considers how modes are used in practice (*e.g.* HTTPS)
  - Many issues in practice because of bad modes!
- This lecture focuses on failures
  - Learn from other's mistakes!

How Not to Use a Blockcipher

- No mode of operation (or ECB)
- Repeated nonces
- Predictable IVs (CBC)
- Metadata leaks information
- Encryption without authentication
- Padding oracles
- Metadata not authenticated
- Too much data with the same key

Notations

- E Block-cipher encryption
- n Block size
- κ Key size
- $\mathcal{E}$ Mode of operation
- M Plaintext $M = m_0 \| m_1 \| \dots$
- C Ciphertext $C = c_0 \| c_1 \| \dots$
- S Secret to recover

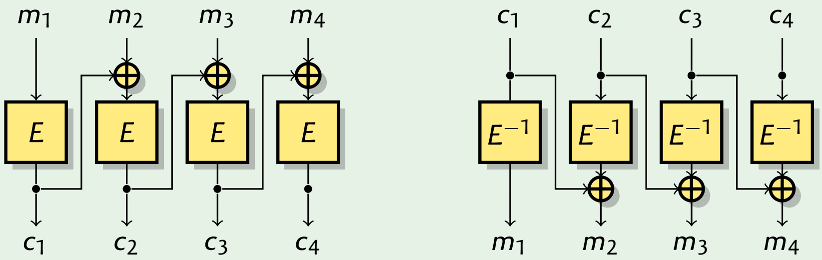
Outline

Outline

Modes of operation

- Encryption must be dependant on the position of the block
  - Use chaining rule
- Non-deterministic to encrypt several messages with the same key
  - Use a different Initialization Value (IV) for each message

Cipher Block Chaining (CBC)



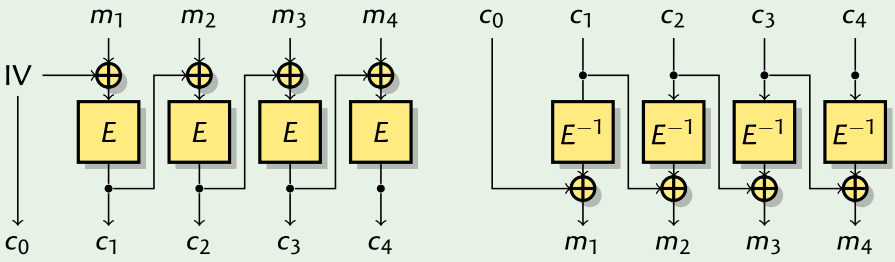
$$c_i = E(m_i \oplus c_{i-1})$$

$$m_i = E^{-1}(c_i) \oplus c_{i-1}$$

Modes of operation

- Encryption must be dependant on the position of the block
  - Use chaining rule
- Non-deterministic to encrypt several messages with the same key
  - Use a different Initialization Value (IV) for each message

Cipher Block Chaining (CBC)

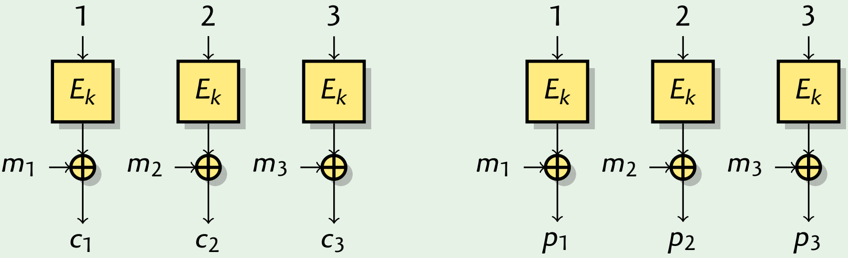


$c_i = E(m_i \oplus c_{i-1})$ $m_i = E^{-1}(c_i) \oplus c_{i-1}$

Modes of operation

- Alternatively, we can use a block-cipher to build a stream-cipher
  - Generate a key-steam z_i
  - Encryption: $c_i = m_i \oplus z_i$
- Different IV for different messages

Counter mode (CTR)



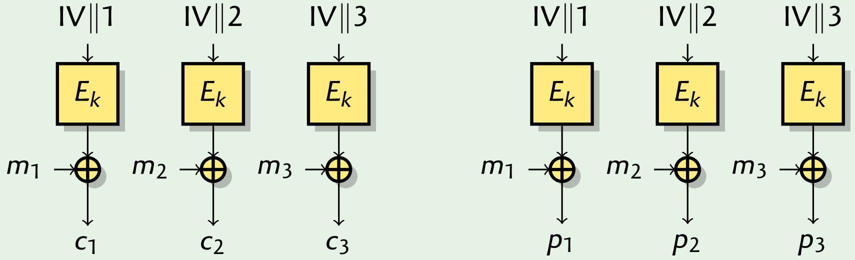
$c_i = m_i \oplus E(i)$

$m_i = c_i \oplus E(i)$

Modes of operation

- Alternatively, we can use a block-cipher to build a stream-cipher
  - Generate a key-steam z_i
  - Encryption: $c_i = m_i \oplus z_i$
- Different IV for different messages

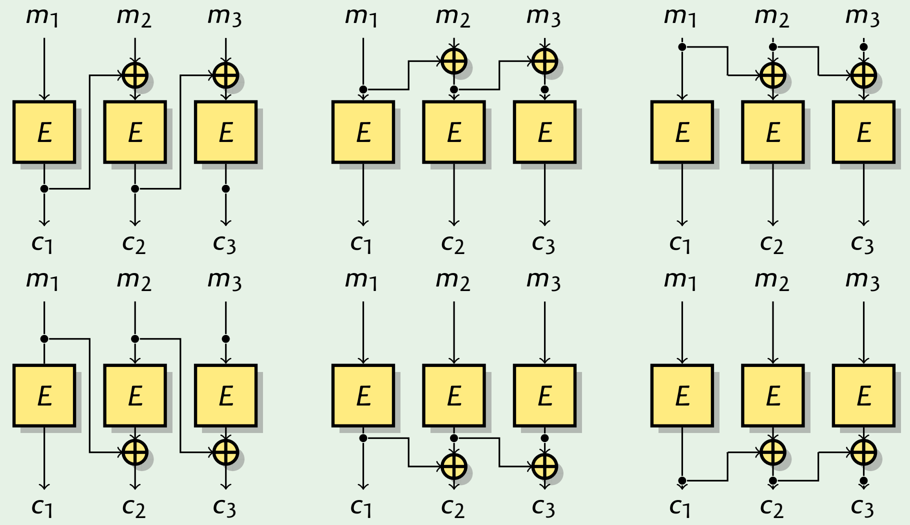
Counter mode (CTR)



$c_i = m_i \oplus E(\mathrm{IV} \| i)$ $\qquad$ $m_i = c_i \oplus E(\mathrm{IV} \| i)$

Chaining rules

Exercise: Which of the following chaining rules are sound?



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:

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

with q the number of queries, σ the total number of blocks

- Proof idea: if inputs to E are distinct, outputs are random
- The CPA security of CBC is essentially the PRP security of E (the block cipher)

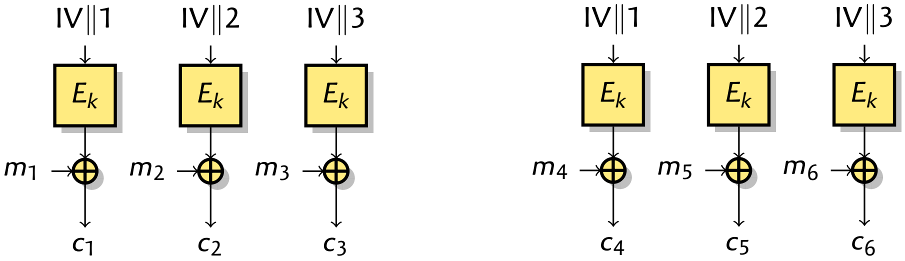
- Many details must be done right for the proof to hold.

IV and nonce

- In CTR, we need the block cipher inputs to be distinct
- Several options:
  1. Stateful counter across messages
  2. Use a random starting point and increment
     - IV must be random
     - Cannot be chosen by adversay
  3. Concatenate IV and counter (reset counter for new message)
     - IV must only be unique: called a nonce
     - Can be chosen by adversary
     - Limits message length

Nonce misuse

- Some errors can lead to repeated IVs
  - Implementation error
  - Weak RNG
  - Random collisions (with short nonces)
- With CTR, this leads to repeated keystream $z_i = z_j$
  - Therefore $c_i \oplus c_j = m_i \oplus m_j$
  - Recover m_j if m_i is known



Nonce misuse

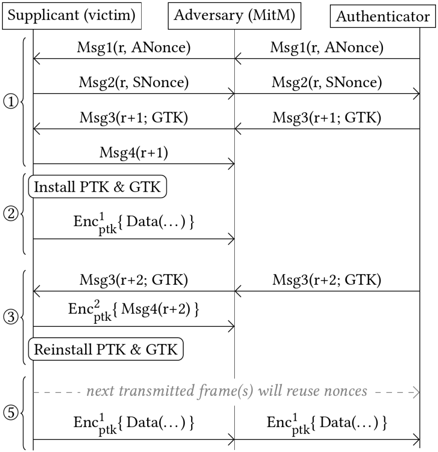
- Some errors can lead to repeated IVs
  - Implementation error
  - Weak RNG
  - Random collisions (with short nonces)
- With CTR, this leads to repeated keystream $z_i = z_j$
  - Therefore $c_i \oplus c_j = m_i \oplus m_j$
  - Recover m_j if m_i is known

Example (WEP)

- WEP uses 24-bit IVs
- Collision expected after $2^{12} = 4096$ messages

Attack in practice: KRACK [Vanhoef & Piessens, 2017]

- Flaw in WPA handshake (WiFi) allows nonce reuse
- Attacker can recover messages with a few queries



How Not to Use a Blockcipher

- No mode of operation (or ECB)
- Repeated nonces
- Predictable IVs (CBC)
- Metadata leaks information
- Encryption without authentication
- Padding oracles
- Metadata not authenticated
- Too much data with the same key

IV in CBC

- Can we use a counter as the IV in CBC?
  - With high probability, $\mathrm{IV} + 1 = \mathrm{IV} \oplus 1$
  - $\mathcal{E}(\mathrm{IV}, m) = \mathrm{IV}, E(m \oplus \mathrm{IV})$
  - $\mathcal{E}(\mathrm{IV} \oplus 1, m \oplus 1) = \mathrm{IV} \oplus 1, E(m \oplus \mathrm{IV})$
- Attack is possible if IV is predictable
  - $\mathcal{E}(\mathrm{IV}_1, m) = \mathrm{IV}_1, E(m \oplus \mathrm{IV}_1)$
  - $\mathcal{E}(\mathrm{IV}_2, m \oplus \mathrm{IV}_1 \oplus \mathrm{IV}_2) = \mathrm{IV}_2, E(m \oplus \mathrm{IV}_1)$
- CBC IV must be random

Exercise: Message recovery in the CPSS model

Can we recover S if the IV is repeated?
Can we recover S if the IV is predictable?

IV in CBC

- Can we use a counter as the IV in CBC?
  - With high probability, $\mathsf{IV}+1 = \mathsf{IV} \oplus 1$
  - $\mathcal{E}(\mathsf{IV}, m) = \mathsf{IV}, E(m \oplus \mathsf{IV})$
  - $\mathcal{E}(\mathsf{IV} \oplus 1, m \oplus 1) = \mathsf{IV} \oplus 1, E(m \oplus \mathsf{IV})$
- Attack is possible if IV is predictable
  - $\mathcal{E}(\mathsf{IV}_1, m) = \mathsf{IV}_1, E(m \oplus \mathsf{IV}_1)$
  - $\mathcal{E}(\mathsf{IV}_2, m \oplus \mathsf{IV}_1 \oplus \mathsf{IV}_2) = \mathsf{IV}_2, E(m \oplus \mathsf{IV}_1)$
- CBC IV must be random

Exercise: Message recovery in the CPSS model

Can we recover S if the IV is repeated?
Can we recover S if the IV is predictable?

IV in CBC

- Can we use a counter as the IV in CBC?
  - With high probability, $\mathrm{IV}+1 = \mathrm{IV} \oplus 1$
  - $\mathcal{E}(\mathrm{IV}, m) = \mathrm{IV}, E(m \oplus \mathrm{IV})$
  - $\mathcal{E}(\mathrm{IV} \oplus 1, m \oplus 1) = \mathrm{IV} \oplus 1, E(m \oplus \mathrm{IV})$
- Attack is possible if IV is predictable
  - $\mathcal{E}(\mathrm{IV}_1, m) = \mathrm{IV}_1, E(m \oplus \mathrm{IV}_1)$
  - $\mathcal{E}(\mathrm{IV}_2, m \oplus \mathrm{IV}_1 \oplus \mathrm{IV}_2) = \mathrm{IV}_2, E(m \oplus \mathrm{IV}_1)$
- CBC IV must be random

Exercise: Message recovery in the CPSS model

Can we recover S if the IV is repeated?
Can we recover S if the IV is predictable?

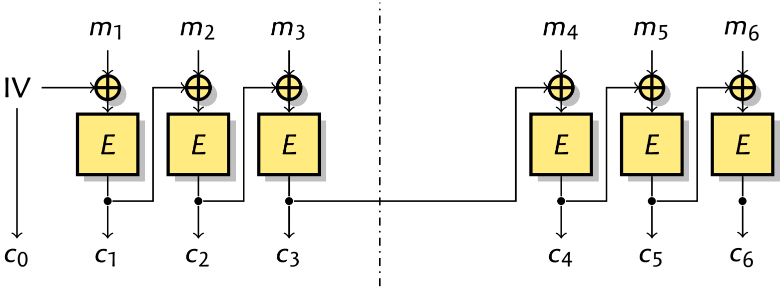
IV in CBC

- Can we use a counter as the IV in CBC?
  - With high probability, $\mathsf{IV} + 1 = \mathsf{IV} \oplus 1$
  - $\mathcal{E}(\mathsf{IV}, m) = \mathsf{IV}, E(m \oplus \mathsf{IV})$
  - $\mathcal{E}(\mathsf{IV} \oplus 1, m \oplus 1) = \mathsf{IV} \oplus 1, E(m \oplus \mathsf{IV})$
- Attack is possible if IV is predictable
  - $\mathcal{E}(\mathsf{IV}_1, m) = \mathsf{IV}_1, E(m \oplus \mathsf{IV}_1)$
  - $\mathcal{E}(\mathsf{IV}_2, m \oplus \mathsf{IV}_1 \oplus \mathsf{IV}_2) = \mathsf{IV}_2, E(m \oplus \mathsf{IV}_1)$
- CBC IV must be random

Exercise: Message recovery in the CPSS model

Can we recover S if the IV is repeated?
Can we recover S if the IV is predictable?

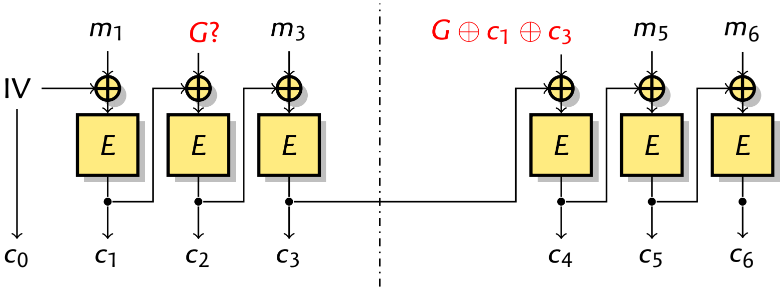
Blockwise-adaptive attacks [Joux & al., Crypto'02]



- CBC Encryption is online
  - Constrained implementations receive m_i, send c_i
  - Attacker can see c_i and adaptively choose m_{i+1}
- TLS 1.0, SSH2: last ciphertext block as IV [Dai, '02], [Rogaway, '02]
  - Attacker can adaptively choose message with known IV

Blockwise-adaptive attacks

[Joux & al., Crypto'02]



1. Make a guess G for m_i
2. After seeing c_{j-1}, sets $m_j = c_{j-1} \oplus G \oplus c_{i-1}$
3. If guess is correct, $c_j = c_i$.

- Message recovery with 256 queries

Attack in practice: BEAST [Duong & Rizzo, '11]

- SSL and TLS 1.0 use the last ciphertext block as IV
  - Known issue since 2002
  - Countermeasure implemented but disabled to interoperability issues
- Difficulty: HTTP requests start with fixed bytes
  - `GET /...`
  - Use plugins/extension to get more control (Java/Websocket/...)
- Introduction of sliding method for plaintext recovery
- Recovery of HTTP cookies: 256 requests per byte

Countermeasure: $1/n-1$ split

- SSL message split as two CBC messages: 1 byte and $n-1$ bytes
- First message: predictable IV, but not enough plaintext
- Second message: unpredictable IV

Attack in practice: BEAST [Duong & Rizzo, '11]

- SSL and TLS 1.0 use the last ciphertext block as IV
  - Known issue since 2002
  - Countermeasure implemented but disabled to interoperability issues
- Difficulty: HTTP requests start with fixed bytes
  - `GET /...`
  - Use plugins/extension to get more control (Java/Websocket/...)
- Introduction of sliding method for plaintext recovery
- Recovery of HTTP cookies: 256 requests per byte

Countermeasure: $1/n-1$ split

- SSL message split as two CBC messages: 1 byte and $n-1$ bytes
- First message: predictable IV, but not enough plaintext
- Second message: unpredictable IV

How Not to Use a Blockcipher

- No mode of operation (or ECB)
- Repeated nonces
- Predictable IVs (CBC)
- Metadata leaks information
- Encryption without authentication
- Padding oracles
- Metadata not authenticated
- Too much data with the same key

Padding

- CBC can only process full blocks
- We use a padding rule

0 *pad* Pad the last block with zero

m_0	m_1	00	00

- Between 0 and $n-1$ bits of padding
- Plaintext length must be transmitted

10^* *pad* Add single 1 bit, and pad with zero

m_0	m_1	80	00

- Between 1 and n bits of padding
- Receiver can decrypt and remove padding

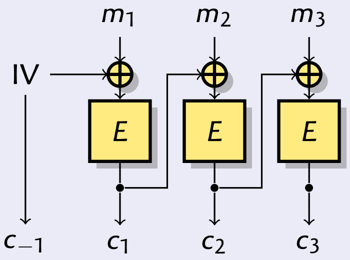
Length pad Last byte is the padding length

m_0	m_1	00	02

- Between 8 and n bits of padding
- Receiver can decrypt and remove padding

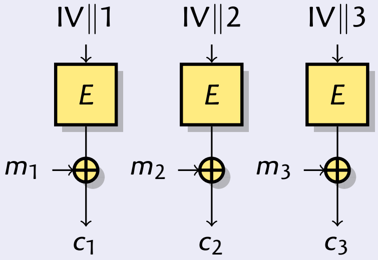
Summary: CBC and CTR mode

CBC mode



- Sequential

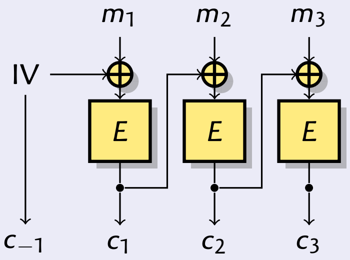
CTR mode



- Parallelizable
- No padding, no expansion
- IV can be a counter
- Blockwise-adaptive security

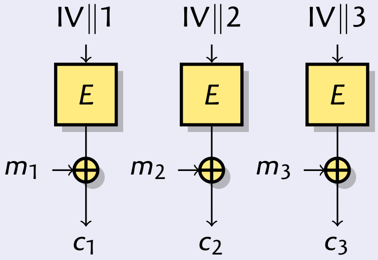
Summary: CBC and CTR mode

CBC mode



- Sequential

CTR mode



- Parallelizable
- No padding, no expansion
- IV can be a counter
- Blockwise-adaptive security

Limitation: Metadata

- Encryption leaks metadata
  - Message length
  - Timings
  - Origin and destination
  - ...

- Sometimes, this is sufficient to find confidential info
  - IP `a.b.c.d` connects to IP of `cancer.org`
  - Wikipedia page with length ℓ, with images of length ℓ_i
  - ...

- NSA collects metadata...

Limitation: Metadata

- Encryption leaks metadata
  - Message length
  - Timings
  - Origin and destination
  - …
- Sometimes, this is sufficient to find confidential info
  - IP `a.b.c.d` connects to IP of `cancer.org`
  - Wikipedia page with length ℓ, with images of length ℓ_i
  - …
- NSA collects metadata…

Limitation: Metadata

- Encryption leaks metadata
  - Message length
  - Timings
  - Origin and destination
  - …
- Sometimes, this is sufficient to find confidential info
  - IP `a.b.c.d` connects to IP of `cancer.org`
  - Wikipedia page with length ℓ, with images of length ℓ_i
  - …
- NSA collects metadata…

Limitation: Metadata

- Encryption leaks metadata
  - Message length
  - Timings
  - Origin and destination
  - …
- Sometimes, this is sufficient to find confidential info
  - IP `a.b.c.d` connects to IP of `cancer.org`
  - Wikipedia page with length ℓ, with images of length ℓ_i
  - …
- NSA collects metadata...

Limitation: Metadata

- Encryption leaks metadata
  - Message length
  - Timings
  - Origin and destination
  - ...
- Sometimes, this is sufficient to find confidential info
  - IP `a.b.c.d` connects to IP of `cancer.org`
  - Wikipedia page with length ℓ, with images of length ℓ_i
  - ...
- NSA collects metadata...

Attack in practice: CRIME [Rizzo & Duong, 2012]

- HTTP, TLS, SPDY support optional compression
  - SPDY has compression by default
- Compression changes length depending on plaintext
  - Leaks information [Kelsey, FSE'02]
- Attacker guesses part of secret, and includes it in message
  - If guess is correct, compression makes the message smaller
  - Message length is visible in ciphertext
  - Recovery of HTTP cookies: 256 requests per byte

Query 1

```
GET /dummy?Cookie: A HTTP/1.1
Cookie: ABCD
```

Query 2

```
GET /dummy?Cookie: B HTTP/1.1
Cookie: ABCD
```

How Not to Use a Blockcipher

- No mode of operation (or ECB)
- Repeated nonces
- Predictable IVs (CBC)
- Metadata leaks information
- Encryption without authentication
- Padding oracles
- Metadata not authenticated
- Too much data with the same key

Limitation: Malleability

- Good encryption: ciphertext indistinguishable from random
  - Adversary learns nothing about plaintext
- Doesn't protect against ciphertext manipulation!

Malleability of CTR

- If c_i is replaced by $c_i' \oplus \delta$, decryption gives $m_i' = m_i \oplus \delta$

$M =$	T	r	a	n	s	f	e	r		$	1	0	0		t	o		B	o	b	.
$C =$	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}	c_{11}	c_{12}	c_{13}	c_{14}	c_{15}	c_{16}	c_{17}	c_{18}	c_{19}	c_{20}	c_{21}
$C' =$	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}	c'_{11}	c_{12}	c_{13}	c_{14}	c_{15}	c_{16}	c_{17}	c_{18}	c_{19}	c_{20}	c_{21}
$M' =$	T	r	a	n	s	f	e	r		$	9	0	0		t	o		B	o	b	.

Limitation: Malleability

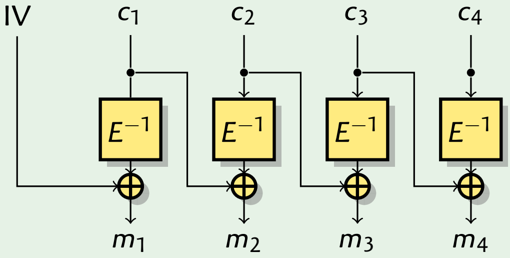
- Good encryption: ciphertext indistinguishable from random
  - Adversary learns nothing about plaintext
- Doesn't protect against ciphertext manipulation!

Exercise: Malleability of CBC

Limitation: Malleability

- Good encryption: ciphertext indistinguishable from random
  - Adversary learns nothing about plaintext
- Doesn't protect against ciphertext manipulation!

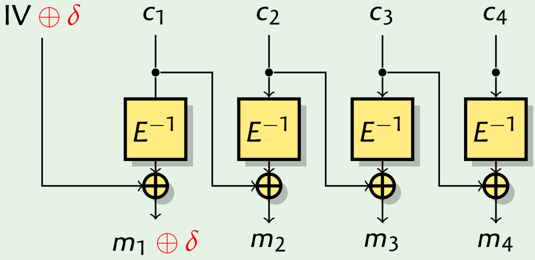
Exercise: Malleability of CBC



Limitation: Malleability

- Good encryption: ciphertext indistinguishable from random
  - Adversary learns nothing about plaintext
- Doesn't protect against ciphertext manipulation!

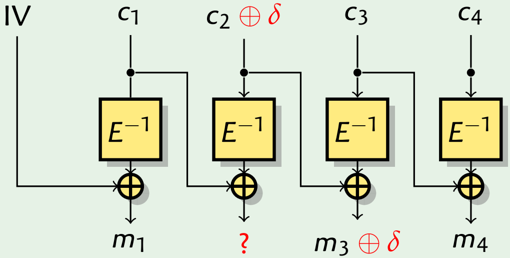
Exercise: Malleability of CBC



Limitation: Malleability

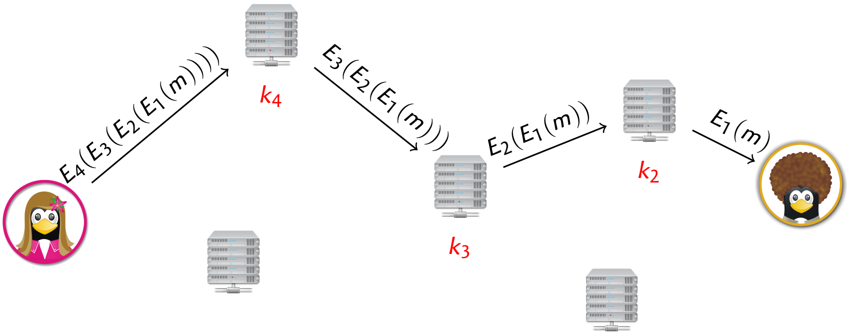
- Good encryption: ciphertext indistinguishable from random
  - Adversary learns nothing about plaintext
- Doesn't protect against ciphertext manipulation!

Exercise: Malleability of CBC



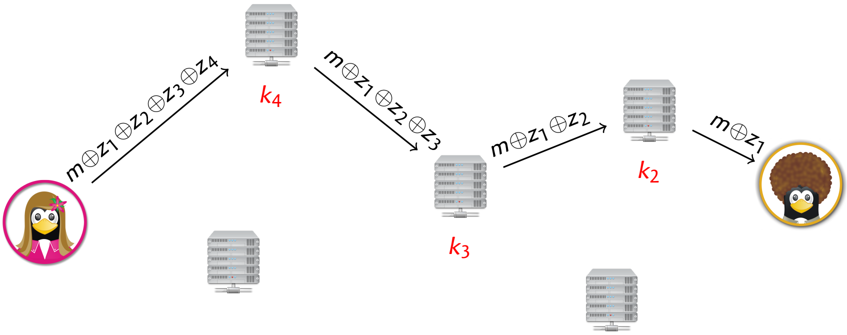
Attack in practice: TOR tagging

- Tor is an anonymity network
  - Packet are encrypted multiple times, and decrypted by each router
- Encryption uses CTR
  - Tagging attack: routers can verify that they are on the same circuit



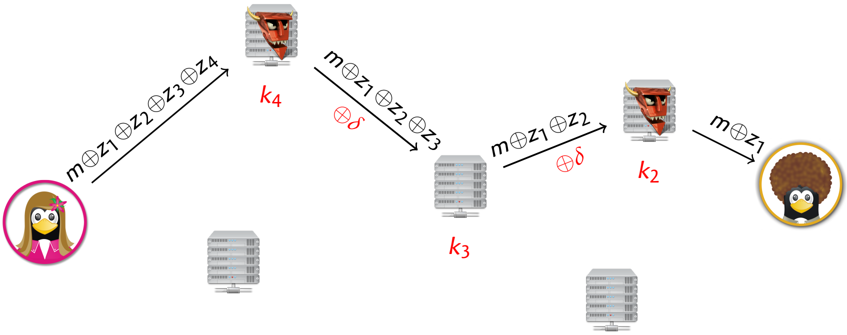
Attack in practice: TOR tagging

- Tor is an anonymity network
  - Packet are encrypted multiple times, and decrypted by each router
- Encryption uses CTR
  - Tagging attack: routers can verify that they are on the same circuit

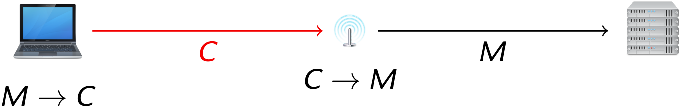


Attack in practice: TOR tagging

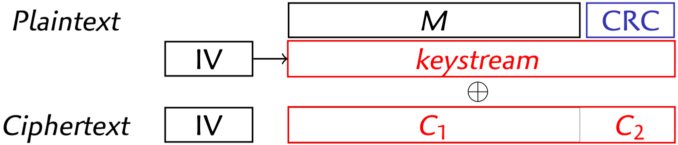
- Tor is an anonymity network
  - Packet are encrypted multiple times, and decrypted by each router
- Encryption uses CTR
  - Tagging attack: routers can verify that they are on the same circuit



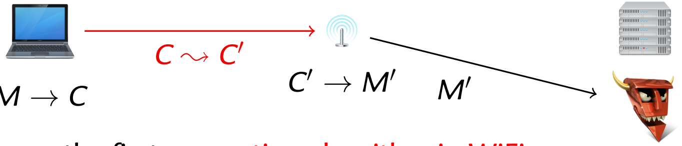
Attack in practice: WEP IP redirection [Borisov & al., 2001]



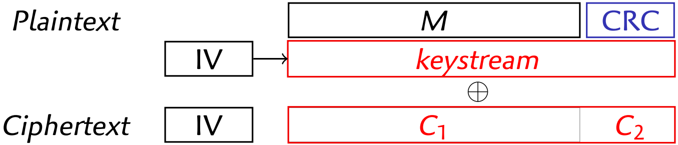
- WEP was the first encryption algorithm in WiFi
- Take message, append CRC, encrypt with stream cipher



Attack in practice: WEP IP redirection [Borisov & al., 2001]



- WEP was the first encryption algorithm in WiFi
- Take message, append CRC, encrypt with stream cipher



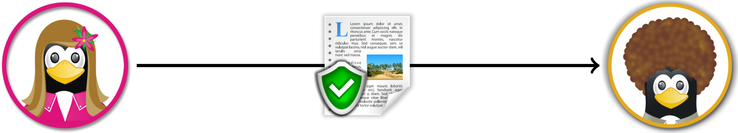
- Problem: Linear CRC does not prevent malleability
  - $\mathrm{CRC}(M \oplus \Delta) = \mathrm{CRC}(M) \oplus \mathrm{CRC}(\Delta)$
  - $C_1' := C_1 \oplus \Delta$
  - $C_2' := C_2 \oplus \mathrm{CRC}(\Delta)$
- Modify IP header: Router decrypt message, sends plaintext to target

How Not to Use a Blockcipher

- No mode of operation (or ECB)
- Repeated nonces
- Predictable IVs (CBC)
- Metadata leaks information
- Encryption without authentication
- Padding oracles
- Metadata not authenticated
- Too much data with the same key

Outline

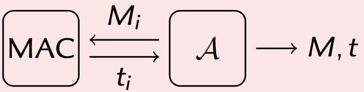
Message Authentication Codes (MAC)



M, t

- Ensures integrity of the message
  - Alice uses a key k to compute a tag: $t = \text{MAC}_k(M)$
  - Bob verifies the tag with the same key k: $t \stackrel{?}{=} \text{MAC}_k(M)$

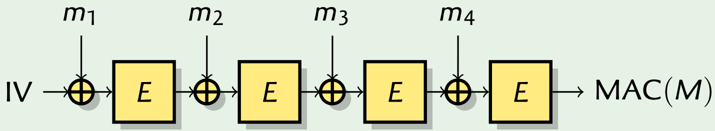
Security notion: forgery



- Adversary makes MAC queries
- Predicts MAC of a new message

CBC-MAC

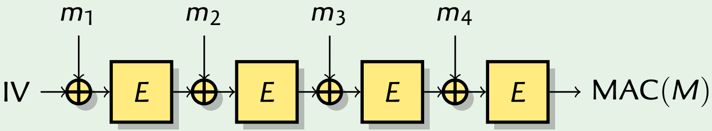
CBC-MAC: first attempt



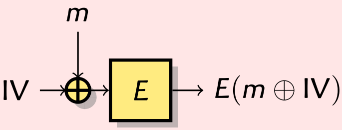
- Last CBC ciphertext block depends on the key and full message
- Can we use it for authentication?

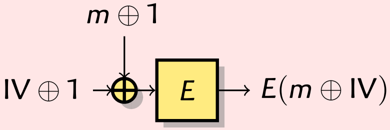
CBC-MAC

CBC-MAC: first attempt



Forgery

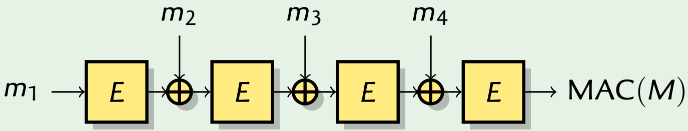




- Query m, get t; Query t get t'
- Forge with $m \| 0, t'$

CBC-MAC

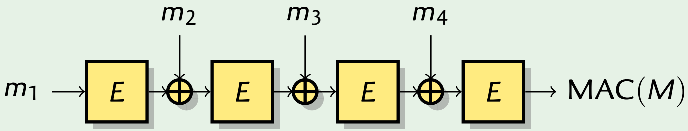
CBC-MAC: second attempt



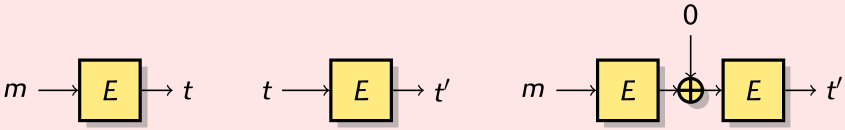
- We don't need an IV for a MAC!
- Is it secure now?

CBC-MAC

CBC-MAC: second attempt



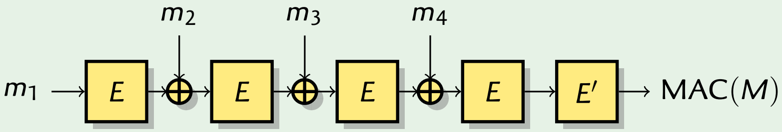
Forgery



- Query m, get t; Query t get t'
- Forge with $m \| 0, t'$

CBC-MAC

CBC-MAC: second attempt



- We need to do something different at the end
  - Encrypt-last-block CBC-MAC: Encrypt with a different key
  - Many variants: FCBC, XCBC, OMAC, ... [Black & Rogaway '00]

Authenticated encryption

- Authenticated encryption combines encryption and MAC to provide confidentiality and authenticity

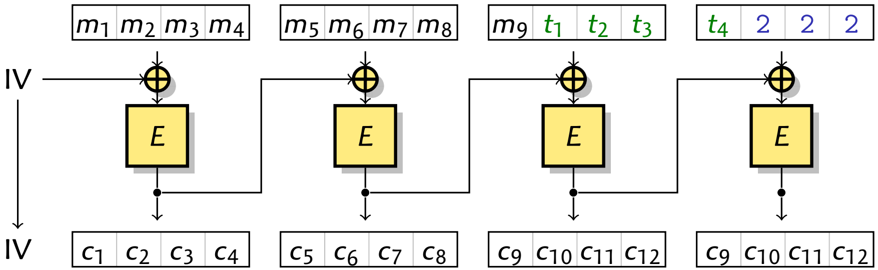
- Different way to combine, some are better than others...
- The keys must be independent

CBC and CBC-MAC with the same key

- CBC plaintext/ciphertext gives input/output pairs for E
- Can be used for forgeries

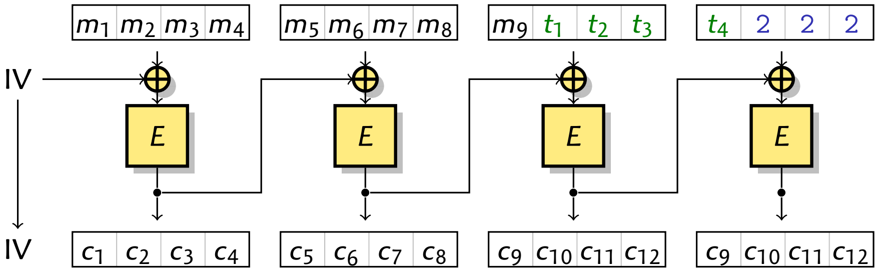
TLS authenticated encryption

1. Compute MAC t of message
2. Concatenate M and t, pad with padding length
3. Encrypt with CBC



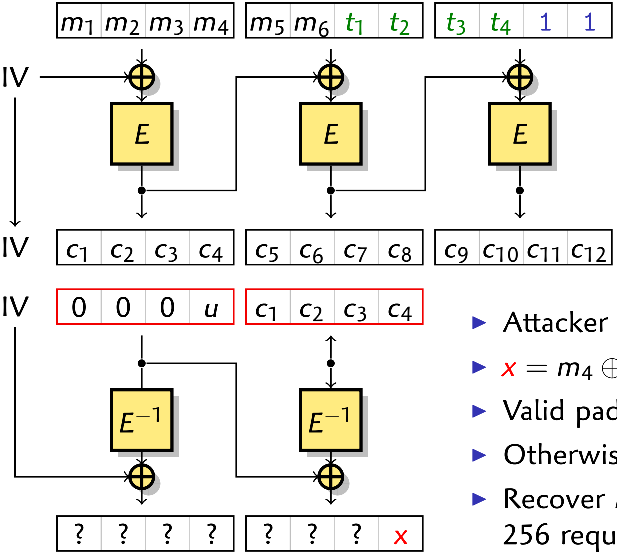
TLS authenticated encryption

1. Compute MAC t of message
2. Concatenate M and t, pad with padding length
3. Encrypt with CBC



- Problem: TLS 1.0 requires different errors for invalid padding and invalid MAC
  - Leaks plaintext information

Padding oracle attack on TLS 1.0 [Vaudenay, Eurocrypt'02]



- Attacker manipulates ciphertext
- $x = m_4 \oplus \mathsf{IV}_4 \oplus u$
- Valid padding if $x = 0$
- Otherwise, likely invalid
- Recover m_4 from error message: 256 requests per byte

Padding oracles on TLS

- Padding oracle: different error messages (TLS 1.0)
- Countermeasure: use RC4: broken
- Countermeasure: same error message (TLS 1.1)

- Padding oracle: if padding invalid, receiver doesn't compute MAC
  - Timing of the error message [Canvel & *al.*, Crypto'03]
- Countermeasure: always compute the MAC

- Padding oracle: un-padded message has different length
  - Timing of the error message [Lucky13, S&P'13]
- Countermeasure: constant-time decryption: hard
- Countermeasure: use GCM (TLS 1.2)

Padding oracles on TLS

- Padding oracle: different error messages (TLS 1.0)
- Countermeasure: use RC4: broken
- Countermeasure: same error message (TLS 1.1)

- Padding oracle: if padding invalid, receiver doesn't compute MAC
  - Timing of the error message [Canvel & *al.*, Crypto'03]
- Countermeasure: always compute the MAC

- Padding oracle: un-padded message has different length
  - Timing of the error message [Lucky13, S&P'13]
- Countermeasure: constant-time decryption: hard
- Countermeasure: use GCM (TLS 1.2)

Padding oracles on TLS

- Padding oracle: different error messages (TLS 1.0)
- Countermeasure: use RC4: broken
- Countermeasure: same error message (TLS 1.1)

- Padding oracle: if padding invalid, receiver doesn't compute MAC
  - Timing of the error message [Canvel & *al.*, Crypto'03]
- Countermeasure: always compute the MAC

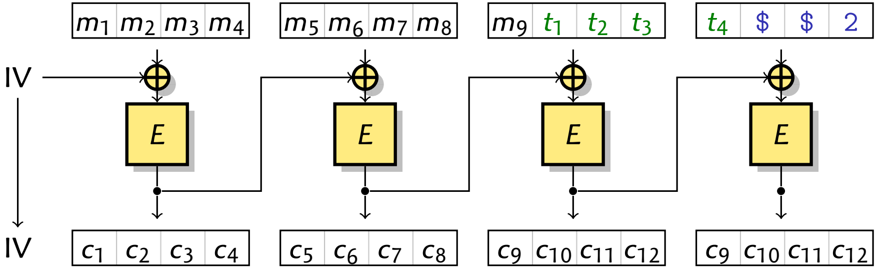
- Padding oracle: un-padded message has different length
  - Timing of the error message [Lucky13, S&P'13]
- Countermeasure: constant-time decryption: hard
- Countermeasure: use GCM (TLS 1.2)

How Not to Use a Blockcipher

- No mode of operation (or ECB)
- Repeated nonces
- Predictable IVs (CBC)
- Metadata leaks information
- Encryption without authentication
- Padding oracles
- Metadata not authenticated
- Too much data with the same key

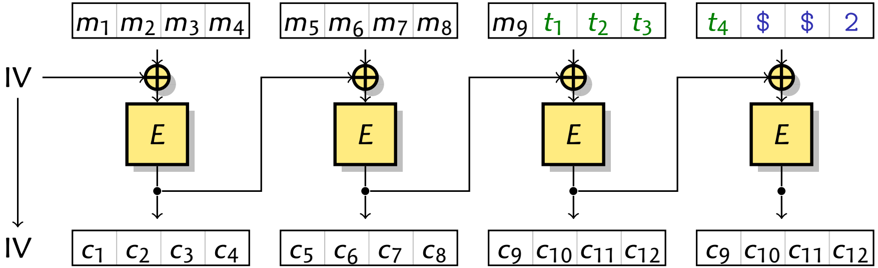
SSL3 authenticated encryption

1. Compute MAC t of message
2. Concatenate M and t, pad with random bytes and padding length
3. Encrypt with CBC



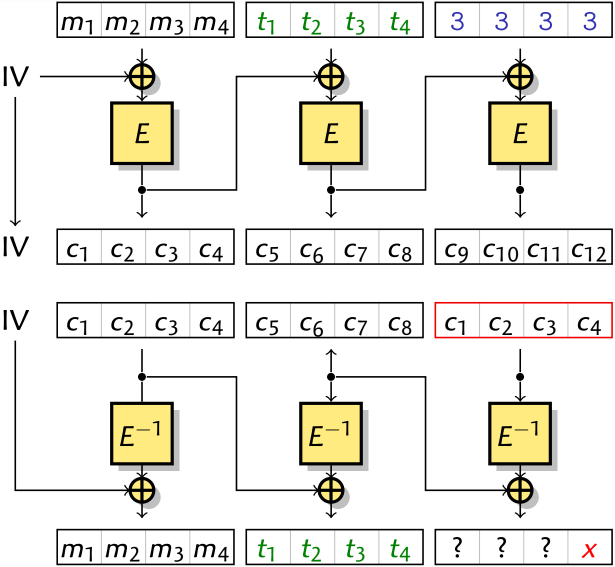
SSL3 authenticated encryption

1. Compute MAC t of message
2. Concatenate M and t, pad with random bytes and padding length
3. Encrypt with CBC



- Problem: padding bytes not authenticated

Padding oracle attack on SSL [POODLE]



- $\mathcal{A}$ manipulates C, observes error, recovers M

- $x = m_4 \oplus \mathrm{IV}_4 \oplus c_8$
- Valid padding and MAC if $x = 3$
- Otherwise, message rejected
- Recover m_4: 256 requests

How Not to Use a Blockcipher

- No mode of operation (or ECB)
- Repeated nonces
- Predictable IVs (CBC)
- Metadata leaks information
- Encryption without authentication
- Padding oracles
- Metadata not authenticated
- Too much data with the same key

Outline

Security of modes of operation

- Most modes (CBC, CTR, GCM, ...) have a security proof like:

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

  with q the number of queries, σ the total number of blocks
- 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 ($2^{n/2}$)

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 (circa 2016)

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 (circa 2016)

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.

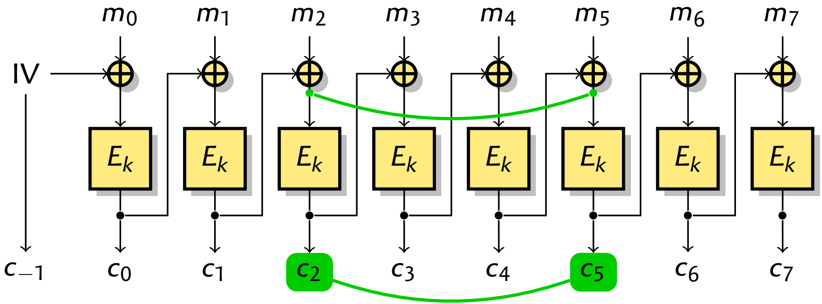
What implementation do (circa 2016)

TLS libraries, web browsers no rekeying

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

CBC collisions

- Well known collision attack against CBC

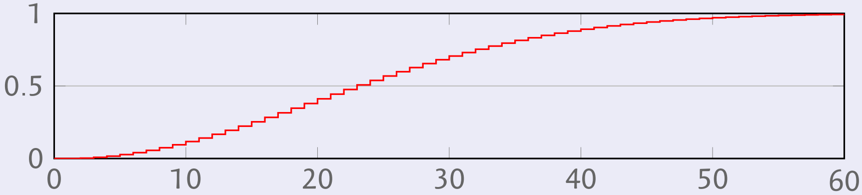


- If $c_i = c_j$, then $c_{i-1} \oplus m_i = c_{j-1} \oplus m_j$
- Ciphertext collision reveals the xor of two plaintext blocks

Birthday paradox

The birthday paradox

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



Security of CBC

- CBC leaks plaintext after $2^{n/2}$ blocks encrypted with the same key
- Security of mode is lower than security of cipher

Birthday paradox

The birthday paradox

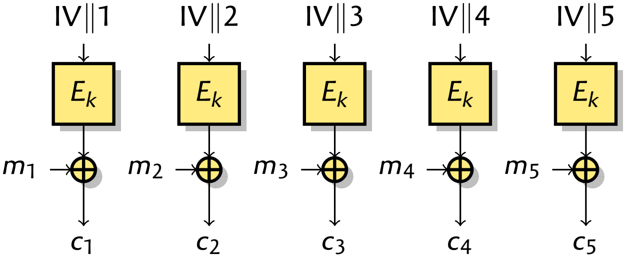
- In a room with 23 people, there is a 50% chance that two of them share the same birthday.
- With random n-bit strings, first collision after roughly $2^{n/2}$ draws.
- More generally, 2^{2t-n} collisions with 2^t draws

Security of CBC

- CBC leaks plaintext after $2^{n/2}$ blocks encrypted with the same key
- Security of mode is lower than security of cipher

Birthday distinguishing on CTR

- Well known distinguisher against CTR

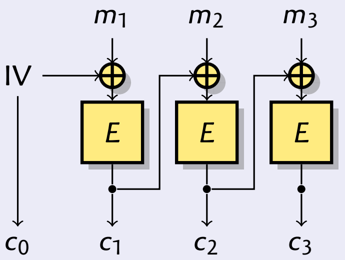


- All block cipher input are distinct
- For all $i \neq j$, $m_i \oplus c_i \neq m_j \oplus c_j$
  - Hard to extract plaintext information from inequalities
- Distinguisher: collision after $2^{n/2}$ blocks with random ciphertext

CBC vs. CTR

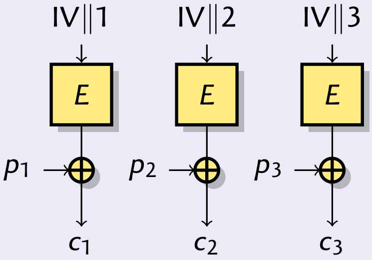
CBC mode

- Collisions reveals
  xor of two plaintext blocks



CTR mode

- Distinguishing attack:
  Key stream doesn't collide



CBC vs. CTR

CBC mode

- Collisions reveals xor of two plaintext blocks

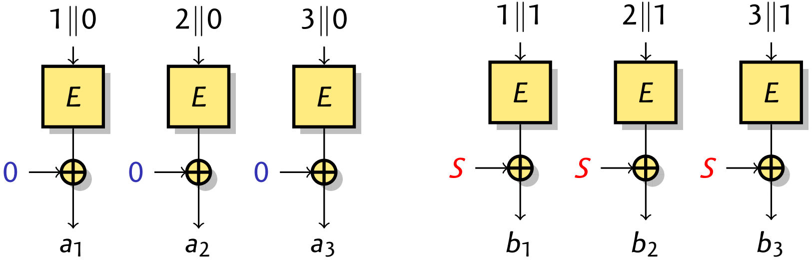
CTR mode

- Distinguishing attack: Key stream doesn't collide

Cryptography engineering [Ferguson, Schneier, Kohno]

CTR leaks very little data. [...] It would be reasonable to limit the cipher mode to 2^{60} blocks, which allows you to encrypt 2^{64} bytes but restricts the leakage to a small fraction of a bit.
When using CBC mode you should be a bit more restrictive. [...] We suggest limiting CBC encryption to 2^{32} blocks or so.

Plaintext recovery on CTR

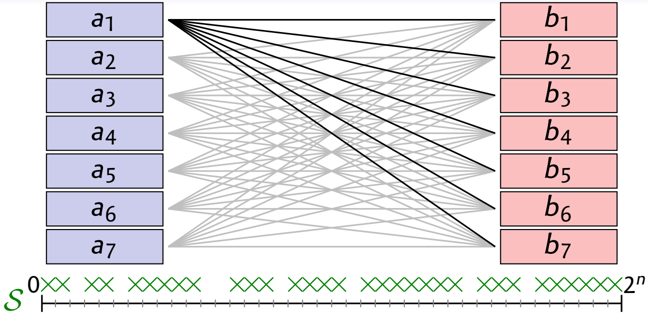


Missing difference problem

- Collect two kind of blocks
  - $a_i = E(i)$
  - $b_j = E(j) \oplus S$
- $\forall i, j,\ S \neq a_i \oplus b_j$

Sieving algorithm

[McGrew, FSE'13]



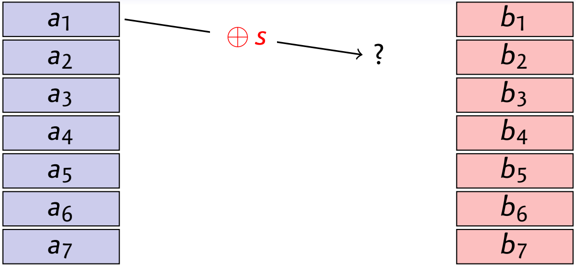
- Compute all $a_i \oplus b_j$, remove from a sieve $\mathcal{S}$

Analysis: Coupon collector problem

- To exclude 2^n candidates S, we need $n \cdot 2^n$ values $a_i \oplus b_j$
  - Lists $\mathcal{A}$ and $\mathcal{B}$ of size $\sqrt{n} \cdot 2^{n/2}$. Complexity: $\tilde{\mathcal{O}}(2^n)$

Searching algorithm

[McGrew, FSE'13]

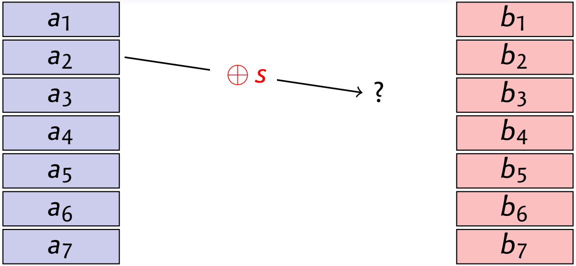


- Make a guess for S, and verify
- With CPSS queries, only 1 unknown byte
  - Complexity: $\tilde{O}(2^{n/2})$

Try Guess

for a **in** $\mathcal{A}$ **do**
  if $(s \oplus a) \in \mathcal{B}$ **then**
    return 0
return 1

Searching algorithm [McGrew, FSE'13]



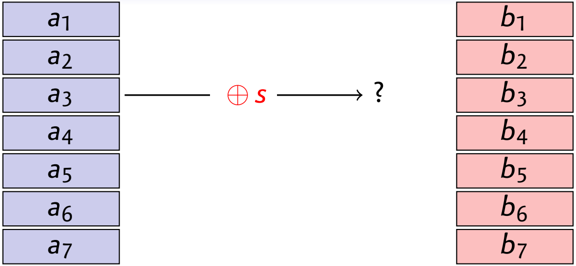
- Make a guess for S, and verify
- With CPSS queries, only 1 unknown byte
  - Complexity: $\tilde{\mathcal{O}}(2^{n/2})$

Try Guess

for a **in** $\mathcal{A}$ **do**
  if $(s \oplus a) \in \mathcal{B}$ **then**
    return 0
return 1

Searching algorithm

[McGrew, FSE'13]



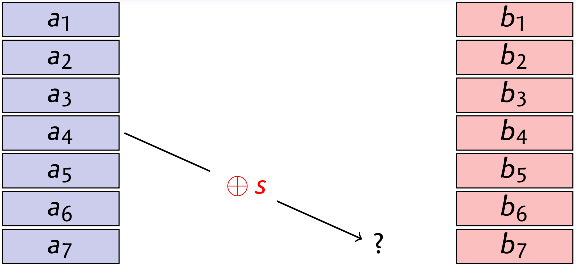
- Make a guess for S, and verify
- With CPSS queries, only 1 unknown byte
  - Complexity: $\tilde{\mathcal{O}}(2^{n/2})$

Try Guess

for a **in** $\mathcal{A}$ **do**
  if $(s \oplus a) \in \mathcal{B}$ **then**
    return 0
return 1

Searching algorithm

[McGrew, FSE'13]



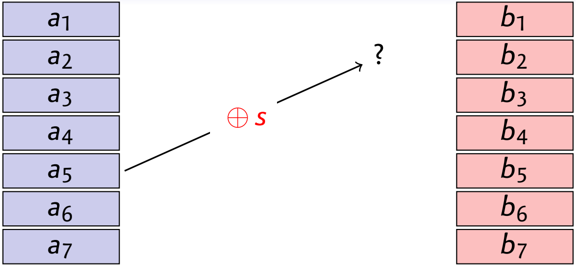
- Make a guess for S, and verify
- With CPSS queries, only 1 unknown byte
  - Complexity: $\tilde{O}(2^{n/2})$

Try Guess

for a **in** $\mathcal{A}$ **do**
  if $(s \oplus a) \in \mathcal{B}$ **then**
    return 0
return 1

Searching algorithm

[McGrew, FSE'13]



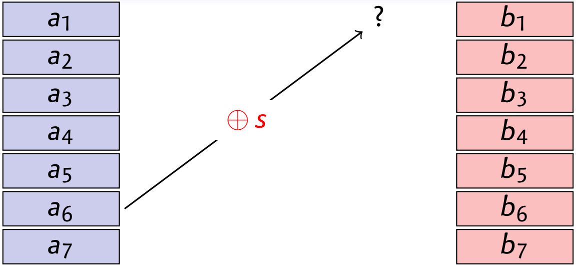
- Make a guess for S, and verify
- With CPSS queries, only 1 unknown byte
  - Complexity: $\tilde{O}(2^{n/2})$

Try Guess

for a **in** $\mathcal{A}$ **do**
  if $(s \oplus a) \in \mathcal{B}$ **then**
    return 0
return 1

Searching algorithm

[McGrew, FSE'13]



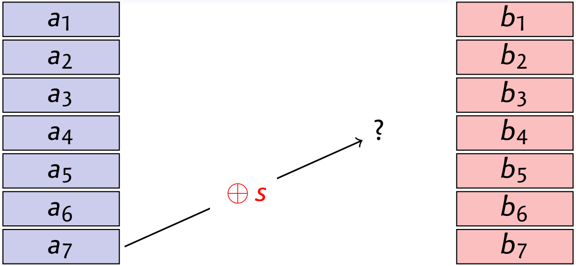
- Make a guess for S, and verify
- With CPSS queries, only 1 unknown byte
  - Complexity: $\tilde{O}(2^{n/2})$

Try Guess

for a **in** $\mathcal{A}$ **do**
  if $(s \oplus a) \in \mathcal{B}$ **then**
    return 0
return 1

Searching algorithm

[McGrew, FSE'13]



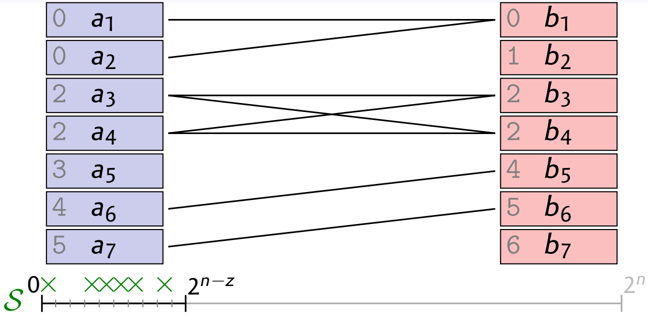
- Make a guess for S, and verify
- With CPSS queries, only 1 unknown byte
  - Complexity: $\tilde{O}(2^{n/2})$

Try Guess

for a **in** $\mathcal{A}$ **do**
  if $(s \oplus a) \in \mathcal{B}$ **then**
    return 0
return 1

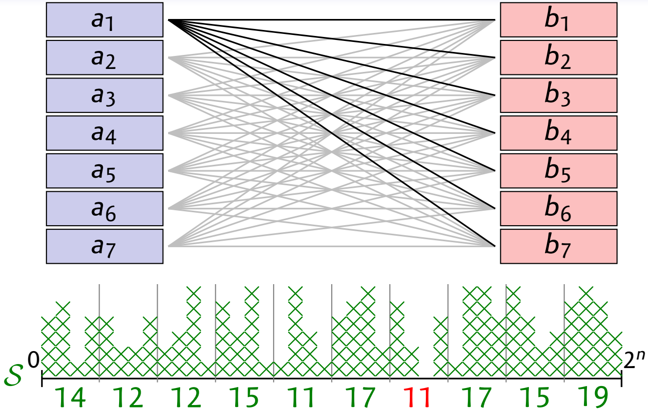
Known-prefix sieving

[L & Sibleyras, EC'18]



- Assume S starts with z zero bits (*e.g.* CPSS queries)
  - Smaller sieve
- Sort lists, consider a_i's and b_j's with matching prefix
- Complexity: $\tilde{\mathcal{O}}(2^{n/2})$ when $z \geq n/2$

Fast Convolution Sieving [L & Sibleyras, EC'18]



- Use $2^{2n/3}$ queries, sieving with $2^{2n/3}$ buckets of $2^{n/3}$ elements
  - With high probability, missing difference has smallest buckets
- Sieving can be computed with Fast Walsh-Hadamard transform!
  - Complexity: $\tilde{\mathcal{O}}(2^{2n/3})$ for arbitrary S

CBC vs. CTR

CBC mode

- Collisions reveals xor of two plaintext blocks

CTR mode

- Distinguishing attack: Key stream doesn't collide
- Message recovery attack with birthday complexity

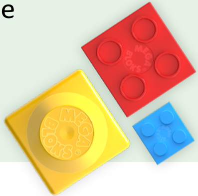
Cryptography engineering [Ferguson, Schneier, Kohno]

CTR leaks very little data. [...] It would be reasonable to limit the cipher mode to 2^{60} blocks, which allows you to encrypt 2^{64} bytes but restricts the leakage to a small fraction of a bit.
When using CBC mode you should be a bit more restrictive. [...] We suggest limiting CBC encryption to 2^{32} blocks or so.

Block size in practice

Block size is an important security parameter

- Block ciphers from the 90's have a 64-bit block size
  - Blowfish, DES, 3DES
- Modern block ciphers have a 128-bit block size
  - AES, Twofish, CAMELLIA

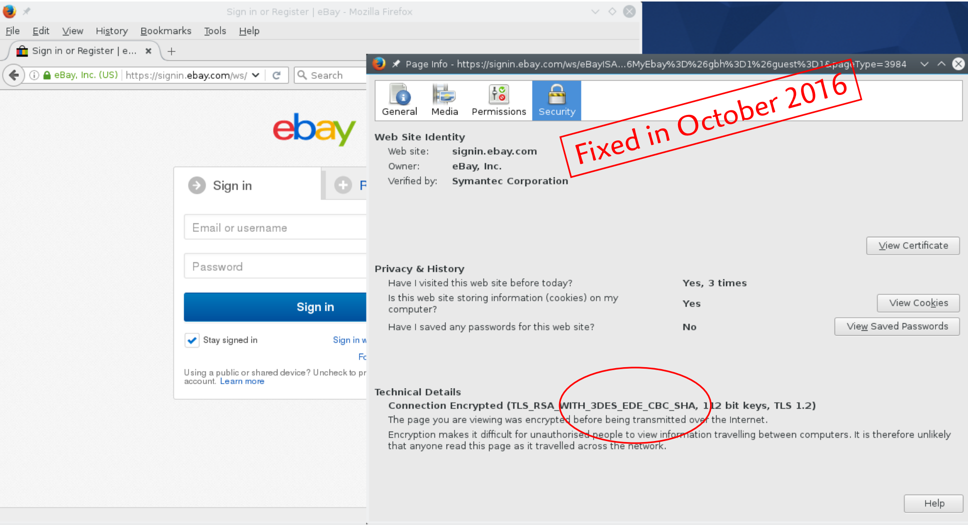


- With $n = 64$, the birthday bound is only 32 GB
- Around 1—2% of HTTPS connections use 3DES-CBC

	February 2016		October 2016		January 2017	
3DES	support	use	support	use	support	use
Top 1k	93%	1.6%	84%	1.5%	75%	1.1%
Top 1M	86%	1.3%	86%	1.0%	76%	0.8%

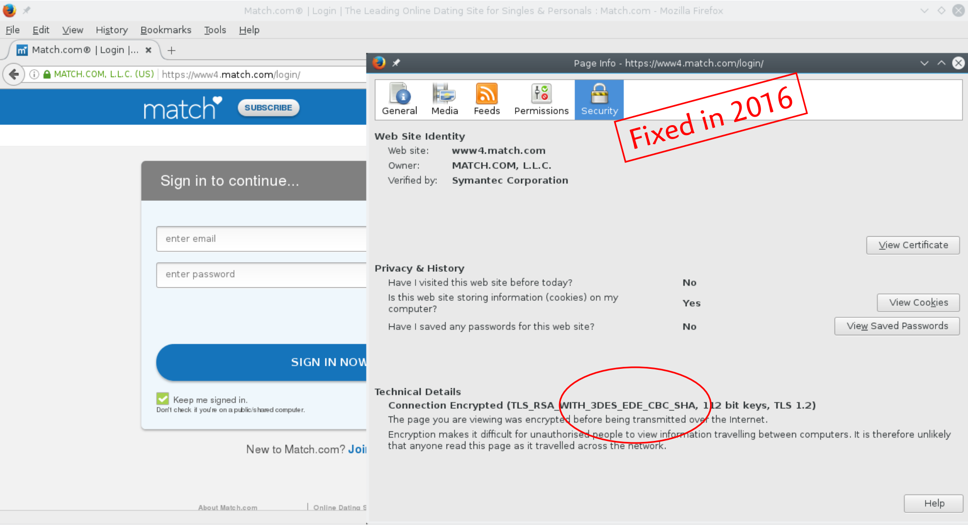
Poorly configured websites

ebay.com



Poorly configured websites

match.com



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

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

Disabled in 2016

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_RC2_128_CBC_EXPORT40_WITH_MD5 SSL 2.0

Attack in practice: Sweet32

[Bhargavan & L, CCS'16]

2^t

Plaintext

$2^{n/2-t/2}$ 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

Attack in practice: Sweet32

[Bhargavan & L, CCS'16]

2^t

Plaintext		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
$2^{n/2-t/2}$	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

Attack in practice: Sweet32

[Bhargavan & L, CCS'16]

2^t

Plaintext		GET	␣/i	nde	x.h	tml	␣HT	TP/	1.1	Coo	kie	:␣C	=??	???
$2^{n/2-t/2}$ Ciphertexts	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

Attack in practice: Sweet32

[Bhargavan & L, CCS'16]

2^t

Plaintext

$2^{n/2-t/2}$ 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

Attack in practice: Sweet32

[Bhargavan & L, CCS'16]

2^t

$2^{n/2-t/2}$ Ciphertexts

Plaintext		GET	␣/i	nde	x.h	tml	␣HT	TP/	1.1	Coo	kie	:␣C	=??	???
Ciphertexts	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

Attack in practice: Sweet32

[Bhargavan & L, CCS'16]

2^t

Plaintext	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

$2^{n/2-t/2}$ Ciphertexts

Attack in practice: Sweet32

[Bhargavan & L, CCS'16]

2^t

Plaintext	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

$2^{n/2-t/2}$ Ciphertexts

Attack in practice: Sweet32

[Bhargavan & L, CCS'16]

2^t

Plaintext		GET	␣/i	nde	x.h	tml	␣HT	TP/	1.1	Coo	kie	:␣C	=??	???
$2^{n/2-t/2}$ Ciphertexts	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

Attack in practice: Sweet32

[Bhargavan & L, CCS'16]

2^t

Plaintext

$2^{n/2-t/2}$ 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

Attack in practice: Sweet32

[Bhargavan & L, CCS'16]

2^t

Plaintext	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

$2^{n/2-t/2}$ *Ciphertexts*

Attack in practice: Sweet32

[Bhargavan & L, CCS'16]

2^t

Plaintext

$2^{n/2-t/2}$ 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

Attack in practice: Sweet32

[Bhargavan & L, CCS'16]

2^t

Plaintext

$2^{n/2-t/2}$ Ciphertexts

Plaintext	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

Attack in practice: Sweet32 [Bhargavan & L, CCS'16]

2^t

Plaintext

$2^{n/2-t/2}$ Ciphertexts

Plaintext	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

Proof-of-concept Attack Demo

- 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
2. Capture on the network with `tcpdump`
3. Remove header, extract ciphertext at fixed position
4. Sort ciphertext (`stdxxl`), look for collisions

- Expected time: 38 hours for 785 GB (tradeoff q. size / # q.).
- In practice: 30.5 hours for 610 GB.

Another target

OpenVPN uses Blowfish-CBC by default

Disclosure

Sweet32 attack disclosed on August 24

- `https://sweet32.info`
- CVE-2016-2183, CVE-2016-6329

- OpenVPN 2.4 has cipher negotiation defaulting to AES
- Mozilla has implemented data limits in Firefox 51 (1M records)

Block size does matter

- Birthday attack against CBC with $2^{n/2}$ data
- Protocols from the 90's still use 64-bit ciphers
- Attacks with 2^{32} data are practical

How Not to Use a Blockcipher

- No mode of operation (or ECB)
- Repeated nonces
- Predictable IVs (CBC)
- Metadata leaks information
- Encryption without authentication
- Padding oracles
- Metadata not authenticated
- Too much data with the same key

Conclusion

- It's easy to make mistakes
  - Mistakes in widely used protocols: SSL, TLS, SSH, WEP, WPA, ...
- Pay attention to security assumptions
  - Security model
  - Nonces/IV
  - ...
- Distinguisher matters
  - They can often be turned into real attacks
  - Protocols should be fixed as soon as issue are found