How Not to Use a Blockcipher

Gaëtan Leurent

Inria

COST Training School, Feb. 2018
Alice and Bob want to communicate securely
- Confidentiality
- Integrity

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

They’ve heard about block ciphers... How to use them?
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 $2^k$ 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: $M = m_1 \parallel m_2 \parallel \ldots$
- Encrypt block independently: $C = E(m_1) \parallel E(m_2) \parallel \ldots$
A block cipher is a family of permutations
- Should behave like a set of $2^\kappa$ 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: $M = m_1 \parallel m_2 \parallel \ldots$
- Encrypt block independently: $C = E(m_1) \parallel E(m_2) \parallel \ldots$

**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
  - ...
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 cryptanalytic 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:

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

```javascript
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]

1. **Attacker** has access to the network *(e.g., public WiFi)*

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

3. **Attacker** captures encrypted data

4. **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 $M \mapsto E(M|S)$
  - Secret included in the message

Exercise: Message recovery

Can we recover $S$ in this model with ECB encryption?
We can model these attacks as Chosen-Prefix Secret-Suffix

- Fixed secret high-value $S$
- Oracle access $M \mapsto E(M||S)$
  - Secret included in the message

**Exercise: Message recovery**

Can we recover $S$ in this model with ECB encryption?
We can model these attacks as Chosen-Prefix Secret-Suffix

- Fixed secret high-value $S$
- Oracle access $M \mapsto E(M\|S)$
  - 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 $\ell = n - 1 \mod n$
   Guess last block: single unknown byte
   
   $\begin{array}{cccccccc}
   0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 2 & 0 & 0 & 3 \\
   \end{array}$ ... $\begin{array}{cccccc}
   0 & 0 & 0 & s_0 & s_1 & s_2 & s_3 \\
   \end{array}$

2. When guess is correct, collision reveals $s_0$

3. Use a prefix of length $\ell = n - 2 \mod n$:
   
   $\begin{array}{cccccccc}
   0 & 0 & s_0 & 0 & 0 & s_0 & 1 & 0 & 0 & s_0 & 2 & 0 & 0 & s_0 & 3 \\
   \end{array}$ ... $\begin{array}{cccccc}
   0 & 0 & s_0 & s_1 & s_2 & s_3 \\
   \end{array}$

4. Collision reveals $s_1$

5. Iterate ...
Chosen-Prefix Secret-Suffix

[Hoang & al., Crypto’15]

1. Use a prefix of length $\ell = n - 1 \mod n$
   Guess last block: single unknown byte
   
   \[
   \begin{array}{cccccccc}
   0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 2 & 0 & 0 & 3 & \cdots & 0 & 0 & 0 & S_0 & S_1 & S_2 & S_3 & -
   \end{array}
   \]

2. When guess is correct, collision reveals $s_0$

3. Use a prefix of length $\ell = n - 2 \mod n$:
   
   \[
   \begin{array}{cccccccc}
   0 & 0 & s_0 & 0 & 0 & s_0 & 1 & 0 & 0 & s_0 & 2 & 0 & 0 & s_0 & 3 & \cdots & 0 & 0 & s_0 & s_1 & s_2 & s_3 & - & -
   \end{array}
   \]

4. Collision reveals $s_1$

5. Iterate …
**Chosen-Prefix Secret-Suffix**  

[Hoang & al., Crypto’15]

1. Use a prefix of length $\ell = n - 1 \mod n$  
   Guess last block: single unknown byte
   \[
   \begin{array}{cccccccc}
   0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 2 & 0 & 0 & 3 & \cdots & 0 & 0 & 0 & 0 & s_0 & s_1 & s_2 & s_3 & -
   \end{array}
   \]

2. When guess is correct, collision reveals $s_0$

3. Use a prefix of length $\ell = n - 2 \mod n$:
   \[
   \begin{array}{cccccccc}
   0 & 0 & s_0 & 0 & 0 & 0 & s_0 & 1 & 0 & 0 & s_0 & 2 & 0 & 0 & s_0 & 3 & \cdots & 0 & 0 & s_0 & s_1 & s_2 & s_3 & - & -
   \end{array}
   \]

4. Collision reveals $s_1$

5. Iterate …
Chosen-Prefix Secret-Suffix

1. Use a prefix of length $\ell = n - 1 \mod n$
   Guess last block: single unknown byte
   
   \[
   \begin{array}{cccccccc}
   0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 3 & \cdots & 0 & 0 & 0 & s_0 & s_1 & s_2 & s_3 & -
   \end{array}
   \]

2. When guess is correct, collision reveals $s_0$

3. Use a prefix of length $\ell = n - 2 \mod n$:
   
   \[
   \begin{array}{cccccccc}
   0 & 0 & s_0 & 0 & 0 & s_0 & 1 & 0 & 0 & s_0 & 2 & 0 & 0 & s_0 & 3 & \cdots & 0 & 0 & s_0 & s_1 & s_2 & s_3 & - & -
   \end{array}
   \]

4. Collision reveals $s_1$

5. Iterate \ldots
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
- $\kappa$: Key size
- $\mathcal{E}$: Mode of operation
- $M$: Plaintext $M = m_0\|m_1\|\ldots$
- $C$: Ciphertext $C = c_0\|c_1\|\ldots$
- $S$: Secret to recover
Outline

Introduction

Encryption
  CBC and CTR
  IVs and nonces
  Padding
  Limitations

Authentication
  CBC-MAC
  Authenticated Encryption

Birthday attacks
  CBC
  CTR
  In practice: Sweet32

Conclusion
Outline

Introduction

Encryption

CBC and CTR
IVs and nonces
Padding
Limitations

Authentication

CBC-MAC
Authenticated Encryption

Birthday attacks

CBC
CTR
In practice: Sweet32

Conclusion
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)**

\[
\begin{align*}
  c_i &= E(m_i \oplus c_{i-1}) \\
  m_i &= E^{-1}(c_i) \oplus c_{i-1}
\end{align*}
\]
Modes of operation

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

Counter mode (CTR)

\[ c_i = m_i \oplus E(IV \| i) \]

Similarly:

\[ m_i = c_i \oplus E(IV \| i) \]
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:

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

with \( q \) the number of queries, \( \sigma \) 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.
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 adversary
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

Diagram:

<table>
<thead>
<tr>
<th>IV∥1</th>
<th>IV∥2</th>
<th>IV∥3</th>
</tr>
</thead>
<tbody>
<tr>
<td>( m_1 )</td>
<td>( m_2 )</td>
<td>( m_3 )</td>
</tr>
<tr>
<td>( c_1 )</td>
<td>( c_2 )</td>
<td>( c_3 )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>IV∥1</th>
<th>IV∥2</th>
<th>IV∥3</th>
</tr>
</thead>
<tbody>
<tr>
<td>( m_4 )</td>
<td>( m_5 )</td>
<td>( m_6 )</td>
</tr>
<tr>
<td>( c_4 )</td>
<td>( c_5 )</td>
<td>( c_6 )</td>
</tr>
</tbody>
</table>
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**

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

**Diagram:**

1. **Msg1(r, ANonce)**
   - **Supplicant (victim)**
   - **Adversary (MitM)**
   - **Authenticator**

2. **Install PTK & GTK**
   - **Enc\textsuperscript{1}_{ptk}\{ Data(…) \}**

3. **Enc\textsuperscript{2}_{ptk}\{ Msg4(r+2) \}**

4. **Reinstall PTK & GTK**

5. **next transmitted frame(s) will reuse nonces**

   - **Enc\textsuperscript{1}_{ptk}\{ Data(…) \}**
   - **Enc\textsuperscript{1}_{ptk}\{ Data(…) \}**
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
Can we use a counter as the IV in CBC?

- With high probability, $\text{IV} + 1 = \text{IV} \oplus 1$
- $E(\text{IV}, m) = \text{IV}, E(m \oplus \text{IV})$
- $E(\text{IV} \oplus 1, m \oplus 1) = \text{IV} \oplus 1, E(m \oplus \text{IV})$

Attack is possible if IV is predictable

- $E(\text{IV}_1, m) = \text{IV}_1, E(m \oplus \text{IV}_1)$
- $E(\text{IV}_2, m \oplus \text{IV}_1 \oplus \text{IV}_2) = \text{IV}_2, E(m \oplus \text{IV}_1)$

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, $IV + 1 = IV \oplus 1$
  - $E(IV, m) = IV, E(m \oplus IV)$
  - $E(IV \oplus 1, m \oplus 1) = IV \oplus 1, E(m \oplus IV)$

- Attack is possible if IV is predictable
  - $E(IV_1, m) = IV_1, E(m \oplus IV_1)$
  - $E(IV_2, m \oplus IV_1 \oplus IV_2) = IV_2, E(m \oplus IV_1)$

- 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, $\text{IV} + 1 = \text{IV} \oplus 1$
  - $E(\text{IV}, m) = \text{IV}, E(m \oplus \text{IV})$
  - $E(\text{IV} \oplus 1, m \oplus 1) = \text{IV} \oplus 1, E(m \oplus \text{IV})$

- Attack is possible if IV is predictable
  - $E(\text{IV}_1, m) = \text{IV}_1, E(m \oplus \text{IV}_1)$
  - $E(\text{IV}_2, m \oplus \text{IV}_1 \oplus \text{IV}_2) = \text{IV}_2, E(m \oplus \text{IV}_1)$

- 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, $\text{IV} + 1 = \text{IV} \oplus 1$
  - $E(\text{IV}, m) = \text{IV}, E(m \oplus \text{IV})$
  - $E(\text{IV} \oplus 1, m \oplus 1) = \text{IV} \oplus 1, E(m \oplus \text{IV})$

- Attack is possible if IV is predictable
  - $E(\text{IV}_1, m) = \text{IV}_1, E(m \oplus \text{IV}_1)$
  - $E(\text{IV}_2, m \oplus \text{IV}_1 \oplus \text{IV}_2) = \text{IV}_2, E(m \oplus \text{IV}_1)$

- 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: \( \frac{1}{n} - 1 \) 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

- SSL and TLS 1.0 use the last ciphertext block as IV
  - Known issue since 2002
  - Countermeasure implemented but disabled due 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
  - Between 0 and \( n - 1 \) bits of padding
  - Plaintext length must be transmitted

10* pad  Add single 1 bit, and pad with zero
  - Between 1 and \( n \) bits of padding
  - Receiver can decrypt and remove padding

Length pad  Last byte is the padding length
  - Between 8 and \( n \) bits of padding
  - Receiver can decrypt and remove padding
Summary: CBC and CTR mode

**CBC mode**

- **IV**
- **m1** → **c1**
- **m2** → **c2**
- **m3** → **c3**

- Sequential

**CTR mode**

- **IV∥1**
- **IV∥2**
- **IV∥3**

- 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 $\ell$, with images of length $\ell_i$
  - ...

- 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 $\ell$, with images of length $\ell_i$
  - ...

- 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 $\ell$, with images of length $\ell_i$
  - ...

- 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 $\ell$, with images of length $\ell_i$
  - ...

- 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 $\ell$, with images of length $\ell_i$
  - ...

- NSA collects metadata...
Attack in practice: CRIME

- 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 $m'_i = m_i \oplus \delta$

$$
M = \text{Transfer} \ $1000 \ \text{to} \ \text{Bob}.
$$
$$
C = \begin{array}{ccccccccccccccccccc}
c_1 & c_2 & c_3 & c_4 & c_5 & c_6 & c_7 & c_8 & c_9 & c_{10} & c_{11} & c_{12} & c_{13} & c_{14} & c_{15} & c_{16} & c_{17} & c_{18} & c_{19} & c_{20} & c_{21} \\
\end{array}
$$
$$
C' = \begin{array}{ccccccccccccccccccc}
c_1 & c_2 & c_3 & c_4 & c_5 & c_6 & c_7 & c_8 & c_9 & c_{10} & c_{11} & c_{12} & c_{13} & c_{14} & c_{15} & c_{16} & c_{17} & c_{18} & c_{19} & c_{20} & c_{21} \\
\end{array}
$$
$$
M' = \text{Transfer} \ $9000 \ \text{to} \ \text{Bob}.
$$
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

\[
E_4(E_3(E_2(E_1(m)))) \quad E_3(E_2(E_1(m))) \quad E_2(E_1(m)) \quad E_1(m)
\]
**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

\[ m \oplus z_1 \oplus z_2 \oplus z_3 \oplus z_4 \]
**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

```
Plaintext  IV  M  CRC
keystream
Ciphertext IV  C1  C2
```

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

\[
\begin{align*}
M &\rightarrow C \\
C &\sim C' \\
C' &\rightarrow M' \\
M' &\rightarrow WEP
\end{align*}
\]

- Problem: Linear CRC does not prevent malleability
  - \[\text{CRC}(M \oplus \Delta) = \text{CRC}(M) \oplus \text{CRC}(\Delta)\]
  - \[C'_1 := C_1 \oplus \Delta\]
  - \[C'_2 := C_2 \oplus \text{CRC}(\Delta)\]

- 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

**Introduction**

**Encryption**
- CBC and CTR
- IVs and nonces
- Padding
- Limitations

**Authentication**
- CBC-MAC
- Authenticated Encryption

**Birthday attacks**
- CBC
- CTR
- In practice: Sweet32

**Conclusion**
Message Authentication Codes (MAC)

- 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** \( k \):
    \[ t \overset{?}{=} \text{MAC}_k(M) \]

**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**

\[
\begin{align*}
\text{IV} &\quad \oplus \quad E & \quad \oplus \quad E & \quad \oplus \quad E & \quad \oplus \quad E \quad \rightarrow \quad \text{MAC}(M)
\end{align*}
\]

**Forgery**

\[
\begin{align*}
\text{IV} &\quad \oplus \quad E \quad \rightarrow \quad E(m \oplus \text{IV}) \\
\text{IV} \oplus 1 & \quad \oplus \quad E \quad \rightarrow \quad E(m \oplus \text{IV})
\end{align*}
\]

- Query \( m \), get \( t \); Query \( t \) get \( t' \)
- Forge with \( m \| 0, t' \)
We don’t need an IV for a MAC!

Is it secure now?
**CBC-MAC**

**CBC-MAC: second attempt**

\[ m_1 \rightarrow E \rightarrow m_2 \rightarrow E \rightarrow m_3 \rightarrow E \rightarrow m_4 \rightarrow E \rightarrow \text{MAC}(M) \]

**Forgery**

- Query \( m \), get \( t \); Query \( t \) get \( t' \)
- Forge with \( m \parallel 0, t' \)
CBC-MAC

CBC-MAC: second attempt

$m_1 \rightarrow E \rightarrow m_2 \rightarrow E \rightarrow m_3 \rightarrow E \rightarrow m_4 \rightarrow E' \rightarrow \text{MAC}(M)$

▶ 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

$\begin{align*}
\text{IV} & \quad \text{IV} \\
| m_1 | m_2 | m_3 | m_4 | & \quad | m_5 | m_6 | m_7 | m_8 | \\
& \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \ quadratic expansion
**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]

$m_1 m_2 m_3 m_4 \oplus m_5 m_6 t_1 t_2 \oplus t_3 t_4 1 1$

Attacker manipulates ciphertext

$x = m_4 \oplus IV_4 \oplus u$

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

![Diagram of SSL3 authenticated encryption]

- $m_1 m_2 m_3 m_4$
- $m_5 m_6 m_7 m_8$
- $m_9 t_1 t_2 t_3$
- $t_4 \$ \$ 2$
- IV
- $c_1 c_2 c_3 c_4$
- $c_5 c_6 c_7 c_8$
- $c_9 c_{10} c_{11} c_{12}$
- $c_9 c_{10} c_{11} c_{12}$
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]

A manipulates $C$, observes error, recovers $M$

$x = m_4 \oplus IV_4 \oplus c_8$

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

Introduction

Encryption
  CBC and CTR
  IVs and nonces
  Padding
  Limitations

Authentication
  CBC-MAC
  Authenticated Encryption

Birthday attacks
  CBC
  CTR
  In practice: Sweet32

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

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

with $q$ the number of queries, $\sigma$ 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** $\sigma \ll 2^{n/2}$

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

\[ \text{IV} \parallel 1 \quad \text{IV} \parallel 2 \quad \text{IV} \parallel 3 \quad \text{IV} \parallel 4 \quad \text{IV} \parallel 5 \]

\[ \begin{align*}
E_k \quad m_1 \quad c_1 \\
E_k \quad m_2 \quad c_2 \\
E_k \quad m_3 \quad c_3 \\
E_k \quad m_4 \quad c_4 \\
E_k \quad m_5 \quad c_5
\end{align*} \]

- 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 mode

- Collisions reveals xor of two plaintext blocks

- IV
  - $m_1$
  - $m_2$
  - $m_3$

- $c_0$, $c_1$, $c_2$, $c_3$

### CTR mode

- Distinguishing attack:
  - Key stream doesn’t collide

- IV
  - $c_1$, $c_2$, $c_3$
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

\[ \begin{align*}
1 \| 0 & \quad E \quad a_1 \\
2 \| 0 & \quad E \quad a_2 \\
3 \| 0 & \quad E \quad a_3 \\
1 \| 1 & \quad E \quad b_1 \\
2 \| 1 & \quad E \quad b_2 \\
3 \| 1 & \quad E \quad b_3 \\
\end{align*} \]

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

Compute all $a_i \oplus b_j$, remove from a sieve $S$

Analysis: Coupon collector problem

To exclude $2^n$ candidates $S$, we need $n \cdot 2^n$ values $a_i \oplus b_j$

Lists $A$ and $B$ of size $\sqrt{n} \cdot 2^{n/2}$. Complexity: $\tilde{O}(2^n)$
### Searching algorithm

![Diagram showing a searching algorithm with variables and operations.]

- 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 A do
    if (s ⊕ a) ∈ B then
        return 0
    return 1
```
Searching algorithm

Make a guess for $S$, and verify

With CPSS queries, only 1 unknown byte

Complexity: $\tilde{O}(2^{n/2})$
Searching algorithm

[McGrew, FSE’13]

\[
\begin{array}{c}
\begin{array}{c}
a_1 \\
a_2 \\
a_3 \\
a_4 \\
a_5 \\
a_6 \\
a_7 \\
\end{array} \\
\oplus s \\
\end{array} \rightarrow \begin{array}{c}
\begin{array}{c}
b_1 \\
b_2 \\
b_3 \\
b_4 \\
b_5 \\
b_6 \\
b_7 \\
\end{array}
\end{array}
\]

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

Try Guess

```plaintext
for a in A do
    if $(s \oplus a) \in B$ then
        return 0
    return 1
```
**Searching algorithm**

[McGrew, FSE’13]

<table>
<thead>
<tr>
<th>$a_1$</th>
<th>$b_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a_2$</td>
<td>$b_2$</td>
</tr>
<tr>
<td>$a_3$</td>
<td>$b_3$</td>
</tr>
<tr>
<td>$a_4$</td>
<td>$b_4$</td>
</tr>
<tr>
<td>$a_5$</td>
<td>$b_5$</td>
</tr>
<tr>
<td>$a_6$</td>
<td>$b_6$</td>
</tr>
<tr>
<td>$a_7$</td>
<td>$b_7$</td>
</tr>
</tbody>
</table>

$\oplus s \quad ?$

- 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 A do
    if (s \oplus a) \in B then
        return 0
    return 1
```
**Searching algorithm**

Make a guess for $S$, and verify

With CPSS queries, only 1 unknown byte

- Complexity: $\tilde{O}(2^{n/2})$

**Try Guess**

```plaintext
for a in A do
  if (s ⊕ a) ∈ B then
    return 0
  return 1
```
Searching algorithm

[McGrew, FSE’13]

a_1
a_2
a_3
a_4
a_5
a_6
a_7
⊕ s

b_1
b_2
b_3
b_4
b_5
b_6
b_7

Make a guess for S, and verify

With CPSS queries, only 1 unknown byte

Complexity: \(\tilde{O}(2^{n/2})\)

Try Guess

\[
\text{for } a \text{ in } \mathcal{A} \text{ do}
\]
\[
\text{if } (s \oplus a) \in \mathcal{B} \text{ then}
\]
\[
\text{return 0}
\]
\[
\text{return 1}
\]
Searching algorithm

- 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 A do
    if (s ⊕ a) ∈ B then
        return 0
    end if
return 1
```
Known-prefix sieving

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: $\tilde{O}(2^{n/2})$ when $z \geq n/2$
Fast Convolution Sieving

- 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: $\tilde{O}(2^{2n/3})$ 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**

<table>
<thead>
<tr>
<th></th>
<th>February 2016</th>
<th>October 2016</th>
<th>January 2017</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>3DES</strong></td>
<td><strong>support</strong></td>
<td><strong>use</strong></td>
<td><strong>support</strong></td>
</tr>
<tr>
<td></td>
<td><strong>use</strong></td>
<td></td>
<td><strong>use</strong></td>
</tr>
<tr>
<td>Top 1k</td>
<td>93%</td>
<td>1.6%</td>
<td>84%</td>
</tr>
<tr>
<td>Top 1M</td>
<td>86%</td>
<td>1.3%</td>
<td>86%</td>
</tr>
</tbody>
</table>
Poorly configured websites

*ebay.com*

Fixed in October 2016
Poorly configured websites

match.com

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

Sign in to continue...

enter email

enter password

SIGN IN NOW

New to Match.com? Join

About Match.com | Online Dating Site

Gaëtan Leurent (Inria) How Not to Use a Blockcipher COST Training School, Feb. 2018 59 / 64

Fixed in 2016
Poorly configured websites

match.com

https://discovery.cryptosense.com/analyze/208.83.241.15
Poorly configured websites

webmail.trumporg.com

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

<table>
<thead>
<tr>
<th>TLS (port 443 - HTTP)</th>
<th>Versions</th>
<th>Ciphers</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSL 2.0, TLS 1.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TLS_RSA_WITH_RC4_128_MD5</td>
<td>TLS 1.0</td>
<td></td>
</tr>
<tr>
<td>TLS_RSA_WITH_RC4_128_SHA</td>
<td>TLS 1.0</td>
<td></td>
</tr>
<tr>
<td>TLS_RSA_WITH_3DES_EDE_CBC_SHA</td>
<td>TLS 1.0</td>
<td></td>
</tr>
<tr>
<td>TLS_RSA_WITH_DES_CBC_SHA</td>
<td>TLS 1.0</td>
<td></td>
</tr>
<tr>
<td>TLS_RSA_EXPORT1024_WITH_RC4_56_SHA</td>
<td>TLS 1.0</td>
<td></td>
</tr>
<tr>
<td>TLS_RSA_EXPORT1024_WITH_DES_CBC_SHA</td>
<td>TLS 1.0</td>
<td></td>
</tr>
<tr>
<td>TLS_RSA_EXPORT_WITH_RC4_40_MD5</td>
<td>TLS 1.0</td>
<td></td>
</tr>
<tr>
<td>TLS_RSA_EXPORT_WITH_RC2_CBC_40_MD5</td>
<td>TLS 1.0</td>
<td></td>
</tr>
<tr>
<td>SSL2_RC4_128_WITH_MD5</td>
<td>SSL 2.0</td>
<td></td>
</tr>
<tr>
<td>SSL2_DES_192_EDE3_CBC_WITH_MD5</td>
<td>SSL 2.0</td>
<td></td>
</tr>
<tr>
<td>SSL2_RC2_128_CBC_WITH_MD5</td>
<td>SSL 2.0</td>
<td></td>
</tr>
<tr>
<td>SSL2_DES_64_CBC_WITH_MD5</td>
<td>SSL 2.0</td>
<td></td>
</tr>
<tr>
<td>SSL2_RC4_128_EXPORT40_WITH_MD5</td>
<td>SSL 2.0</td>
<td></td>
</tr>
<tr>
<td>SSL2_RC2_128_CBC_EXPORT40_WITH_MD5</td>
<td>SSL 2.0</td>
<td></td>
</tr>
</tbody>
</table>

**Disabled in 2016**
**Attack in practice: Sweet32**

[Bhargavan & L, CCS’16]

---

**Plaintext**

```
GET /index.html HT TP/1.1 Cookie: ?? ??
```

---

**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]

Plaintext

\[ \text{GET } \underline{/i\text{ndex}}.\text{html} \underline{HT} TP/1.1 \text{Cookie} : \underline{C} = ?? ?? \]

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

Ciphertexts

\[ 2^n/2 - t/2 \]
## Attack in practice: Sweet32

[Bhargavan & L, CCS’16]

<table>
<thead>
<tr>
<th>Plaintext</th>
<th>178 4E5 71A A39 68A 399 7D8 8F0 FEA 902 932 204 85A 969</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>E57 1AA 396 8A3 997 D88 F0F EA9 029 322 048 5A9 6E0 EA4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Ciphertexts</th>
<th>1D6 645 EA2 050 FAE D74 A72 E5C 913 447 3B4 BAA 321 784</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>7A5 322 700 DE3 BA8 7DD 998 040 A8D 9A2 05A EE5 330 9EC</td>
</tr>
<tr>
<td></td>
<td>9BE 78D 350 AF5 327 311 F5B 252 77A C45 49E 2ED 20C 030</td>
</tr>
<tr>
<td></td>
<td>289 597 BED 540 A60 7AF F96 511 AF2 41F 278 D25 400 4EB</td>
</tr>
<tr>
<td></td>
<td>031 ED8 EEB 6CC B5A 440 067 154 AB5 CEE 015 70A 1ED 1B7</td>
</tr>
<tr>
<td></td>
<td>38E 018 41A DEB 970 2D3 97A F0E 45C 94B 251 218 5FB 82A</td>
</tr>
<tr>
<td></td>
<td>417 FF4 81D 00D 49D D9A 841 737 416 BA8 452 AC0 335 793</td>
</tr>
<tr>
<td></td>
<td>21B B07 A20 4F4 C1D B07 2DF 410 340 6AB 0D2 96B CE9 4C9</td>
</tr>
<tr>
<td></td>
<td>536 BDA A93 B85 351 831 763 FA0 E95 E5F 1EE 986 7D5 8C0</td>
</tr>
<tr>
<td></td>
<td>5F5 935 574 21D EE0 1BF 338 6DB DDC F67 090 7F6 8EC A8D</td>
</tr>
</tbody>
</table>
Attack in practice: Sweet32

[Bhargavan & L, CCS’16]

\[
\begin{align*}
\text{Plaintext} & : \text{GET} /\text{index.html} \text{ HT TP/1.1 Cookie: } \text{=?? ??} \\
& \begin{array}{cccccc}
178 & 4E5 & 71A & A39 & 68A & 399 \\
E57 & 1AA & 396 & 8A3 & 997 & D88 \\
1D6 & 645 & EA2 & 050 & FAE & D74 \\
\end{array} \\
& \begin{array}{cccccc}
7A5 & 322 & 700 & DE3 & BA8 & 7DD \\
9BE & 78D & 350 & AF5 & 327 & 311 \\
289 & 597 & BED & 540 & A60 & 7AF \\
\end{array} \\
& \begin{array}{cccccc}
38E & 018 & 41A & DEB & 970 & 2D3 \\
417 & FF4 & 81D & 00D & 49D & D9A \\
21B & B07 & A20 & 4F4 & C1D & B07 \\
\end{array} \\
& \begin{array}{cccccc}
536 & BDA & A93 & B85 & 351 & 831 \\
5F5 & 935 & 574 & 21D & EE0 & 1BF \\
\end{array}
\end{align*}
\]

\[
\begin{align*}
\text{Ciphertexts} & : 2^n / 2 - t / 2 \\
& \begin{array}{cccccc}
8F0 & FEA & 902 & 932 & 204 & 85A \text{ ??} \\
EA9 & 029 & 322 & 048 & 5A9 & 6E0 \\
FAE & D74 & A72 & E5C & 913 & 447 \\
\end{array} \\
& \begin{array}{cccccc}
A8D & 9A2 & 05A & EE5 & 330 & 9EC \\
C45 & 49E & 2ED & 20C & 030 & \text{ ??} \\
41F & 278 & D25 & 400 & 4EB & \text{ ??} \\
\end{array} \\
& \begin{array}{cccccc}
154 & AB5 & CEE & 015 & 70A & 1ED \\
452 & BA8 & 45B & AC0 & 335 & 793 \\
2DF & 410 & 340 & 6AB & 0D2 & 96B \\
\end{array} \\
& \begin{array}{cccccc}
338 & 6DB & DDC & F67 & 090 & 7F6 \\
8EC & A8D & \text{ ??} & \text{ ??} & \text{ ??} & \text{ ??}
\end{array}
\end{align*}
\]
Attack in practice: Sweet32

[Bhargavan & L, CCS’16]

\[2^t\]

\[2^n/2 - t/2\]

Plaintext

\[\text{GET } \underline{\text{i}} \underline{n} \underline{d}e \underline{x} \underline{.} \underline{h}tml \underline{\text{H}}T \underline{T}P/ \text{1.1 Cookie: } \underline{\text{C}} = ??? ???\]

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

Ciphertexts

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]

<table>
<thead>
<tr>
<th>Plaintext</th>
<th>$2^t$</th>
</tr>
</thead>
<tbody>
<tr>
<td>plain text example</td>
<td></td>
</tr>
<tr>
<td><code>GET/index.html</code></td>
<td></td>
</tr>
<tr>
<td><code>TP/1.1 Cookie:C=?? ??</code></td>
<td></td>
</tr>
<tr>
<td>Ciphertexts</td>
<td></td>
</tr>
<tr>
<td><code>GET</code></td>
<td></td>
</tr>
<tr>
<td><code>index.html</code></td>
<td></td>
</tr>
<tr>
<td><code>TP/1.1</code></td>
<td></td>
</tr>
<tr>
<td><code>Cookie:C=?? ??</code></td>
<td></td>
</tr>
<tr>
<td><code>178 4E5 71A A39 68A 399 7D8 8F0 FEA 902 932 204 85A 969</code></td>
<td>$2^n/2-t/2$</td>
</tr>
<tr>
<td><code>E57 1AA 396 8A3 997 D88 F0F EA9 029 322 048 5A9 6E0 EA4</code></td>
<td></td>
</tr>
<tr>
<td><code>1D6 645 EA2 050 FAE D74 A72 E5C 913 447 3B4 BAA 321 784</code></td>
<td></td>
</tr>
<tr>
<td><code>7A5 322 700 DE3 BA8 7DD 998 040 A8D 9A2 05A EE5 330 9EC</code></td>
<td></td>
</tr>
<tr>
<td><code>9BE 78D 350 AF5 327 311 F5B 252 77A C45 49E 2ED 20C 030</code></td>
<td></td>
</tr>
</tbody>
</table>

Gaëtan Leurent (Inria) How Not to Use a Blockcipher COST Training School, Feb. 2018 60 / 64
**Attack in practice: Sweet32**  

[ Bhargavan & L, CCS’16 ]

<table>
<thead>
<tr>
<th>Plaintext</th>
<th>Ciphertexts</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>GET /index.html HT TP/1.1 Cookie:</code></td>
<td><code>?? ??</code></td>
</tr>
<tr>
<td><code>178 4E5 71A A39 68A 399 7D8 8F0 FEA 902 932 204 85A 969</code></td>
<td><code>2^t</code></td>
</tr>
<tr>
<td><code>E57 1AA 396 8A3 997 D88 F0F EA9 029 322 048 5A9 6E0 EA4</code></td>
<td></td>
</tr>
<tr>
<td><code>1D6 645 EA2 050 FAE D74 A72 E5C 913 447 3B4 BAA 321 784</code></td>
<td></td>
</tr>
<tr>
<td><code>7A5 322 700 DE3 BA8 7DD 998 04O A8D 9A2 05A EE5 330 9EC</code></td>
<td></td>
</tr>
<tr>
<td><code>9BE 78D 350 AF5 327 311 F5B 252 77A C45 49E 2ED 20C 030</code></td>
<td></td>
</tr>
<tr>
<td><code>289 597 BED 540 A60 7AF F96 511 AF2 41F 278 D25 400 4EB</code></td>
<td></td>
</tr>
<tr>
<td><code>031 ED8 EEB 6CC B5A 440 067 154 AB5 CEE 015 70A 1ED 1B7</code></td>
<td></td>
</tr>
<tr>
<td><code>38E 018 41A DEB 970 2D3 97A F0E 45C 94B 251 218 5FB 82A</code></td>
<td></td>
</tr>
<tr>
<td><code>417 FF4 81D 00D 49D D9A 841 737 416 BA8 452 AC0 335 793</code></td>
<td></td>
</tr>
<tr>
<td><code>21B B07 A20 4F4 C1D B07 2DF 410 340 6AB 0D2 96B CE9 4C9</code></td>
<td></td>
</tr>
<tr>
<td><code>536 BDA A93 B85 351 831 763 FA0 E95 E5F 1EE 986 7D5 8C0</code></td>
<td></td>
</tr>
<tr>
<td><code>5F5 935 574 21D EE0 1BF 338 6DB DDC F67 090 7F6 8EC A8D</code></td>
<td></td>
</tr>
</tbody>
</table>
### Attack in practice: Sweet32

[ Bhargavan & L, CCS'16 ]

<table>
<thead>
<tr>
<th>Plaintext</th>
<th>Ciphertexts</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET/index.html/HT/1.1/cookie:</td>
<td>031 ED8 EEB 6CC B5A 440 067 154 AB5 CEE 015 70A 1ED 1B7</td>
</tr>
<tr>
<td>178 4E5 71A A39 68A 399 7D8 8F0 FEA 902 932 204 85A 969</td>
<td>38E 018 41A DEB 970 2D3 97A F0E 45C 94B 251 218 5FB 82A</td>
</tr>
<tr>
<td>E57 1AA 396 8A3 997 D88 F0F EA9 029 322 048 5A9 6E0 EA4</td>
<td>417 FF4 81D 00D 49D D9A 841 737 416 BA8 452 AC0 335 793</td>
</tr>
<tr>
<td>1D6 645 EA2 050 FAE D74 A72 E5C 913 447 3B4 BAA 321 784</td>
<td>289 597 BED 540 A60 7AF F96 511 AF2 41F 278 D25 400 4EB</td>
</tr>
<tr>
<td>7A5 322 700 DE3 BA8 7DD 998 040 A8D 9A2 05A EE5 330 9EC</td>
<td>031 ED8 EEB 6CC B5A 440 067 154 AB5 CEE 015 70A 1ED 1B7</td>
</tr>
</tbody>
</table>

\[ t \cdot 2^{n/2 - t/2} \]
**Attack in practice: Sweet32**

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

**[Bhargavan & L, CCS’16]**

---

**Gaëtan Leurent (Inria)**

**How Not to Use a Blockcipher**

**COST Training School, Feb. 2018**
Attack in practice: Sweet32

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

[Benavides & L, CCS'16]

---

**Plaintext**

<table>
<thead>
<tr>
<th>GET</th>
<th>index</th>
<th>x.html</th>
<th>HT</th>
<th>TP/1.1Cookie::C =????</th>
</tr>
</thead>
<tbody>
<tr>
<td>178</td>
<td>4E5</td>
<td>71A</td>
<td>A39</td>
<td>68A</td>
</tr>
<tr>
<td></td>
<td></td>
<td>399</td>
<td>7D8</td>
<td>8F0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FEA</td>
<td>902</td>
<td>932</td>
</tr>
<tr>
<td></td>
<td></td>
<td>204</td>
<td>85A</td>
<td>969</td>
</tr>
<tr>
<td></td>
<td></td>
<td>E57</td>
<td>1AA</td>
<td>396</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8A3</td>
<td>997</td>
<td>D88</td>
</tr>
<tr>
<td></td>
<td></td>
<td>F0F</td>
<td>EA9</td>
<td>029</td>
</tr>
<tr>
<td></td>
<td></td>
<td>322</td>
<td>048</td>
<td>5A9</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6E0</td>
<td>EA4</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1D6</td>
<td>645</td>
<td>EA2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>050</td>
<td>FAE</td>
<td>D74</td>
</tr>
<tr>
<td></td>
<td></td>
<td>A72</td>
<td>E5C</td>
<td>913</td>
</tr>
<tr>
<td></td>
<td></td>
<td>447</td>
<td>3B4</td>
<td>BAA</td>
</tr>
<tr>
<td></td>
<td></td>
<td>321</td>
<td>784</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>7A5</td>
<td>322</td>
<td>700</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DE3</td>
<td>BA8</td>
<td>7DD</td>
</tr>
<tr>
<td></td>
<td></td>
<td>998</td>
<td>040</td>
<td>A8D</td>
</tr>
<tr>
<td></td>
<td></td>
<td>9A2</td>
<td>05A</td>
<td>EE5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>330</td>
<td>9EC</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>9BE</td>
<td>78D</td>
<td>AF5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>350</td>
<td>327</td>
<td>311</td>
</tr>
<tr>
<td></td>
<td></td>
<td>F5B</td>
<td>252</td>
<td>77A</td>
</tr>
<tr>
<td></td>
<td></td>
<td>C45</td>
<td>49E</td>
<td>2ED</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20C</td>
<td>030</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>289</td>
<td>597</td>
<td>BED</td>
</tr>
<tr>
<td></td>
<td></td>
<td>540</td>
<td>A60</td>
<td>7AF</td>
</tr>
<tr>
<td></td>
<td></td>
<td>511</td>
<td>AF2</td>
<td>41F</td>
</tr>
<tr>
<td></td>
<td></td>
<td>278</td>
<td>D25</td>
<td>400</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4EB</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

**Ciphertexts**

<table>
<thead>
<tr>
<th>031</th>
<th>ED8</th>
<th>EEB</th>
<th>6CC</th>
<th>B5A</th>
<th>440</th>
<th>067</th>
<th>154</th>
<th>AB5</th>
<th>CEE</th>
<th>015</th>
<th>70A</th>
<th>1ED</th>
<th>1B7</th>
</tr>
</thead>
<tbody>
<tr>
<td>38E</td>
<td>018</td>
<td>41A</td>
<td>DEB</td>
<td>970</td>
<td>2D3</td>
<td>97A</td>
<td>F0E</td>
<td>45C</td>
<td>94B</td>
<td>251</td>
<td>218</td>
<td>5FB</td>
<td>82A</td>
</tr>
<tr>
<td>417</td>
<td>FF4</td>
<td>81D</td>
<td>00D</td>
<td>49D</td>
<td>D9A</td>
<td>841</td>
<td>737</td>
<td>416</td>
<td>BA8</td>
<td>452</td>
<td>AC0</td>
<td>335</td>
<td>793</td>
</tr>
<tr>
<td>21B</td>
<td>B07</td>
<td>A20</td>
<td>4F4</td>
<td>C1D</td>
<td>B07</td>
<td>2DF</td>
<td>410</td>
<td>340</td>
<td>6AB</td>
<td>0D2</td>
<td>96B</td>
<td>CE9</td>
<td>4C9</td>
</tr>
</tbody>
</table>

---

Gaëtan Leurent (Inria)  How Not to Use a Blockcipher  COST Training School, Feb. 2018  60 / 64
## Attack in practice: Sweet32

[Bhargavan & L, CCS’16]

### Plaintext

<table>
<thead>
<tr>
<th>GET</th>
<th>/index.html</th>
<th>HT</th>
<th>TP/1.1Cookie: _C=?? ???</th>
</tr>
</thead>
<tbody>
<tr>
<td>178</td>
<td>4E5</td>
<td>71A</td>
<td>A39</td>
</tr>
<tr>
<td>E57</td>
<td>1AA</td>
<td>396</td>
<td>8A3</td>
</tr>
<tr>
<td>1D6</td>
<td>645</td>
<td>EA2</td>
<td>050</td>
</tr>
<tr>
<td>7A5</td>
<td>322</td>
<td>700</td>
<td>DE3</td>
</tr>
<tr>
<td>9BE</td>
<td>78D</td>
<td>350</td>
<td>AF5</td>
</tr>
<tr>
<td>289</td>
<td>597</td>
<td>BED</td>
<td>540</td>
</tr>
</tbody>
</table>

### 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 |

---

Gaëtan Leurent (Inria) How Not to Use a Blockcipher COST Training School, Feb. 2018 60 / 64
Attack in practice: Sweet32

[Bhargavan & L, CCS’16]

\[ 2^n/2 - t/2 \]

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