Symmetric encryption and padding oracles

Tom Ristenpart
CS 6431
An example: On-line shopping with TLS

Step 1: Key exchange protocol to share secret $K$

Step 2: Send data via encrypted channel

$Enc(K, \text{“Quantity: 1, CC#: 5415431230123456”})$

TLS uses many cryptographic primitives:

- **key exchange**: hash functions, digital signatures, public key encryption
- **encrypted channel**: symmetric encryption, message authentication

Mechanisms to resist replay attacks, man-in-the-middle attacks, truncation attacks, etc...
A short history of **TLS**

- **SSL ver 2.0** designed by **Hickman** at Netscape, 1994
- **Wagner, Goldberg** break SSL ver 2, 1995
- **Freier, Karlton, Kocher** design SSL ver 3.0
- **Bleichenbacher** breaks RSA PKCS #1 encryption, used in SSL ver 3, 1998
- **TLS ver 1** released as IETF standard, based on SSL 3, **many cryptographers** involved, 1999
- **Vaudenay, Klima et al.** padding attacks, 2001
- **Rogaway** IV re-use insecurity, 2002
- **Brumley, Boneh** remote timing attacks, 2003
- **TLS ver 1.1** released as standard, 2006

... (more attacks and fixes)

**How many cryptographers involved?**

- 1994
- 1995
- 1998
- 1999
- 2001
- 2002
- 2003
- 2006
TLS 1.2 handshake for RSA transport

Amazon customer

Pick random Nc

Check CERT using CA public verification key

Pick random PMS

C <- Enc(pk, PMS)

Bracket notation means contents encrypted

ClientHello, MaxVer, Nc, Ciphers/CompMethods

ServerHello, Ver, Ns, SessionID, Cipher/CompMethod

CERT = (pk of Amazon, signature over it by CA)

CERT <- D(sk, CERT)

Pick random Ns

C

ChangeCipherSpec, { Finished, PRF(MS, “Client finished” || H(transcript)) }

ChangeCipherSpec, { Finished, PRF(MS, “Server finished” || H(transcript’)) }

MS <- PRF(PMS, “master secret” || Nc || Ns )
Primitives used by TLS

CERT = (pk of bank, signature over it)

Digital signatures

C

Public-key encryption (RSA, Diffie-Hellman)

ChangeCipherSpec,
{ Finished, PRF(MS, “Client finished” || H(transcript)) } PRFs
Hash functions

C1

Symmetric encryption

C2
Symmetric encryption: Game plan

- Block ciphers
- IND$ encryption from block ciphers
- Padding oracle attacks
- Authenticated encryption
- Lucky 13 attack against TLS
- Modern viewpoint: all-in-one AE
Block ciphers

E: \{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^n

Security goal: E(K, M) is indistinguishable from random n-bit string for anyone without K
Block cipher security as PRF

\[ E: \{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^n \]

Adversary gets to submit distinct messages to oracle

\[ \begin{align*}
  &M_1 \quad C_1 \\
  &M_2 \quad C_2 \\
  &\vdots \\
\end{align*} \]

PRF Security Game

\[ \text{Fn}(M) \]

If \( b == 1 \) then
\[ C \leftarrow E(K,M) \]

If \( b == 0 \) then
\[ C \leftarrow \emptyset \{0,1\}^n \]

Ret C

\[ b \] is a uniformly sampled bit and \( K \) is uniformly sampled key
Both hidden from adversary

Adversary outputs guess \( b' \) of \( b \)

Wins if \( b' = b \)

Insecure if adversary wins with probability close to 1

Secure if no adversary can get probability more than 1/2

Pseudorandom permutation (PRP) security is same except that compare to random permutation
Block cipher design

• Cryptanalysts have a pretty impressive laundry list of things to avoid
• Design of good block ciphers a rare expertise (visit Europe)

• We gain confidence by continued inability to break ciphers
• DES and AES competitions facilitated this
Symmetric encryption

R signifies fresh per-message random bits.

Key generation

Handling in TLS key exchange

Optional

Correctness: Dec( K, Enc(K, M, R) ) = M with probability 1 over randomness used
Reductionist approach to cryptography

Formal definitions
Scheme semantics
Security

Security proofs (reductions)

Breaking assumptions
Breaking scheme

As long as assumptions holds we believe in security of scheme!

Provable security yields
1) well-defined assumptions and security goals
2) cryptanalysts can focus on assumptions and models

Example:
Attacker can not recover credit card
Can break block cipher

But no one knows how to do this. It’s been studied for a very long time!
Chosen-plaintext SE security (IND$)

Encryption algorithm denoted Enc

Security goal: Enc(K,M) looks like random bit string to attackers that can obtain encryptions of chosen plaintexts
Block cipher modes of operation

How can we build an encryption scheme for arbitrary message spaces out of a block cipher?

Electronic codebook (ECB) mode
Pad message $M$ to $M_1, M_2, M_3, \ldots$ where each block $M_i$ is $n$ bits

Then:

$$
\begin{align*}
&M1 \\
&\downarrow \\
&E_K \\
&\downarrow \\
&C1 \\
&M2 \\
&\downarrow \\
&E_K \\
&\downarrow \\
&C2 \\
&M3 \\
&\downarrow \\
&E_K \\
&\downarrow \\
&C3
\end{align*}
$$
ECB mode is a more complicated looking substitution cipher

Recall our credit-card number example.
ECB: substitution cipher with alphabet n-bit strings instead of digits

Images courtesy of
OTP-like encryption using block cipher

Counter mode (CTR)
Pad message $M$ to $M_1,M_2,M_3,...$ where each is $n$ bits except last
Choose random $n$-bit string IV
Then:

How do we decrypt?

Maybe use less than full $n$ bits of $P_3$
CBC mode

Ciphertext block chaining (CBC)
Pad message $M$ to $M_1, M_2, M_3, \ldots$ where each block $M_i$ is $n$ bits
Choose random $n$-bit string $IV$
Then:

$$
\begin{align*}
&IV \\ &\downarrow \quad \downarrow \quad \downarrow \\
&C_0 \quad E_K \quad E_K \quad E_K \\
&M_1 \quad M_2 \quad M_3 \\
&\uparrow \quad \uparrow \quad \uparrow \\
&\text{C0} \quad \text{C1} \quad \text{C2} \quad \text{C3}
\end{align*}
$$

How do we decrypt?
Theorem (informal). Let A be an successful, efficient IND$ attacker against security of CBC mode. Then there exists a PRF adversary B against E that is efficient and successful.

Reduces analysis now to E and to security definition / model.
Are we done?

Internet

IV, E(K, IV + M)

https://amazon.com
Active security of CBC mode

What about forging a message? For any $D$:
A padding oracle attack

Assume that $M_1 || M_2$ has length $2n-8$ bits

$P$ is one byte of padding that must equal 0x00

Adversary obtains Ciphertext $C_0, C_1, C_2$

$$\text{Dec}(K, C')$$

$M_1' || M_2' || P' = \text{CBC-Dec}(K, C')$

If $P' \neq 0x00$ then
  Return error
Else
  Return ok
A padding oracle attack

Adversary obtains ciphertext $C = C_0, C_1, C_2$

Let $R$ be arbitrary $n$ bits

Assume that $M_1 || M_2$ has length $2n - 8$ bits

$P$ is one byte of padding that must equal 0x00

Low byte of $M_1$ equals $i$

- $R, C_0, C_1$
  - error

- $R, C_0 \oplus 1, C_1$
  - error

- $R, C_0 \oplus 2, C_1$
  - error

...$

- $R, C_0 \oplus i, C_1$
  - ok

Dec($K, C'$) $M_1' || M_2' || P' = CBC-Dec($K, C'$)

If $P' \neq 0x00$ then
  - Return error

Else
  - Return ok
Padding for CBC Mode in TLS

Possible paddings in TLS:

- 00
- 01 01
- 02 02 02
- etc.

“Lengths longer than necessary might be desirable to frustrate attacks on a protocol that are based on analysis of the lengths of exchanged messages.” RFC 5246
Vaudenay’s padding oracle attack

Goal:
Decrypt entire plaintext
Vaudenay’s padding oracle attack

We know that:

$$00 = i + IV[n] + M1[n]$$

Or do we? Could be:

$$01 = i + IV[n] + M1[n]$$

$$01 = IV[n-1] + M1[n-1]$$

Easy to exclude other cases

$$00...00, C1$$

error

$$00...01, C1$$

error

$$00...02, C1$$

error

... 

$$00...i, C1$$

ok

Dec(K, C')

$$M1' = CBC-Dec(K, C')$$

$$(X, plen) \leftarrow \text{lastbyte}(M1')$$

For $$i = 0$$ to padlen do

$$(X, plen') \leftarrow \text{lastbyte}(X)$$

If $$plen' \neq plen$$

Return Error

Return Ok
Vaudenay’s padding oracle attack

We know $M_1[n]$! Let’s get second to last byte.

Solve $j$ to make $M_1'[n] = 01$

\[ 01 = j + IV[n] + M_1[n] \]

Know that:

\[ 01 = i + IV[n-1] + M_1[n-1] \]

Repeat for all $n$ bytes

\[
\begin{align*}
\text{Dec}(K, C') & \\
M_1' &= \text{CBC-Dec}(K, C') \\
(X, \text{plen}) &\leftarrow \text{lastbyte}(M_1') \\
\text{For } i = 0 \text{ to } \text{padlen} \text{ do } & \\
(X, \text{plen}') &\leftarrow \text{lastbyte}(X) \\
\text{If } \text{plen'} \neq \text{plen} & \Rightarrow \text{Return Error} \\
\text{Return Ok} & \\
\end{align*}
\]

\[
\begin{align*}
00...00 \ j, C1 & \quad \text{error} \\
00...01 \ j, C1 & \quad \text{error} \\
00...02 \ j, C1 & \quad \text{error} \\
00...i \ j, C1 & \quad \text{ok} \\
\end{align*}
\]
Chosen ciphertext attacks against CBC

<table>
<thead>
<tr>
<th>Attack</th>
<th>Description</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vaudenay</td>
<td>10’s of chosen ciphertexts, recovers message bits from a ciphertext. Called “padding oracle attack”</td>
<td>2001</td>
</tr>
<tr>
<td>Canvel et al.</td>
<td>Shows how to use Vaudenay’s ideas against TLS</td>
<td>2003</td>
</tr>
<tr>
<td>Degabriele, Paterson</td>
<td>Breaks IPsec encryption-only mode</td>
<td>2006</td>
</tr>
<tr>
<td>Albrecht et al.</td>
<td>Plaintext recovery against SSH</td>
<td>2009</td>
</tr>
<tr>
<td>Duong, Rizzo</td>
<td>Breaking ASP.net encryption</td>
<td>2011</td>
</tr>
<tr>
<td>Jager, Somorovsky</td>
<td>XML encryption standard</td>
<td>2011</td>
</tr>
<tr>
<td>Duong, Rizzo</td>
<td>“Beast” attacks against TLS</td>
<td>2011</td>
</tr>
<tr>
<td>AlFardan, Patterson</td>
<td>Lucky13 attack against TLS</td>
<td>2013</td>
</tr>
</tbody>
</table>
So far

• IND$ and other CPA modes insecure to active ciphertext-manipulation attacks

• *They do not provide confidentiality*
  – But we had proofs of confidentiality?
  – Provable security is only as good as definitions / models

• TLS actually uses a mode that attempted to fix this
The MAC can be thought of as a pseudorandom function (PRF).
During decryption:
Re-run MAC over header + Payload, check the given value

How to implement this?
(Simplified) Early TLS decryption

```plaintext
TLS-Decrypt(K, C)
X = CBC-Dec(K,C)
(X,plen) <- lastbyte(X)
For i = 0 to padlen do
  (X,plen’) <- lastbyte(X)
  If plen’ != plen
    Send PADDING_ERROR message
(X,tag) <- last20bytes(X)
If MAC(K,X) != tag then
  Send MAC_ERROR MESSAGE
Pass X as plaintext to application
```

How do we mount an attack?
First countermeasure

TLS-Decrypt(K, C)
X = CBC-Dec(K, C)
(X, plen) <- lastbyte(X)
For i = 0 to padlen do
  (X, plen’) <- lastbyte(X)
  If plen’ != plen
    Send MAC_ERROR_MESSAGE
(X, tag) <- last20bytes(X)
If MAC(K, X) != tag then
  Send MAC_ERROR MESSAGE
Pass X as plaintext to application

How do we mount an attack?
First countermeasure

TLS-Decrypt(K, C)
X = CBC-Dec(K,C)
(X,plen) <- lastbyte(X)
For i = 0 to padlen do
  (X,plen’) <- lastbyte(X)
  If plen’ != plen
    Compute MAC(K,X)
    Send MAC_ERROR_MESSAGE
(X,tag) <- last20bytes(X)
If MAC(K,X) != tag then
  Send MAC_ERROR_MESSAGE
Pass X as plaintext to application

How do we mount an attack?
Figure 3: OpenSSL TLS median server timings (in hardware cycles) when $P_{14}^* = 0x01$ and $P_{15}^* = 0xFF$. As expected, $\Delta_{15} = 0xFE$ leads to faster processing time.
High level views of building symmetric encryption

- **Theory**
  - One-way function
  - PRPs/PRFs
  - Encryption, MACs
- **90s / 00s**
  - Blockcipher (assumed PRP)
  - Encryption, MACs
  - Authenticated-Encryption
- **Contemporary**
  - Blockcipher (assume PRP)
  - Tweakable PRPs
  - (Robust) Authenticated-Encryption
## Authenticated encryption schemes

<table>
<thead>
<tr>
<th>Attack</th>
<th>Inventors</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>OCB (Offset Codebook)</td>
<td>Rogaway</td>
<td>One-pass</td>
</tr>
<tr>
<td>GCM (Galios Counter Mode)</td>
<td>McGrew, Viega</td>
<td>CTR mode plus specialized MAC</td>
</tr>
<tr>
<td>CWC</td>
<td>Kohno, Viega, Whiting</td>
<td>CTR mode plus Carter-Wegman MAC</td>
</tr>
<tr>
<td>CCM</td>
<td>Housley, Ferguson, Whiting</td>
<td>CTR mode plus CBC-MAC</td>
</tr>
<tr>
<td>EAX</td>
<td>Wagner, Bellare, Rogaway</td>
<td>CTR mode plus OMAC</td>
</tr>
</tbody>
</table>

Robust variants: SIV mode by Rogaway, Shrimpton

Also see Caeser competition for latest and greatest: [http://competitions.cr.yp.to/caesar-submissions.html](http://competitions.cr.yp.to/caesar-submissions.html)
All-in-one notions

**Game REAL\textsubscript{SE}**

\[ K \leftarrow \mathcal{R} \{0, 1\}^k \]
\[ b' \leftarrow \mathcal{R} \mathcal{A}^{Enc, Dec} \]
\[ \text{ret } b' \]

\[ Enc(N, M) \]
\[ C \leftarrow \text{enc}(K, N, M) \]
\[ \text{ret } C \]

\[ Dec(N, C) \]
\[ \text{Ret Dec}(N, C) \]

**Game RAND\textsubscript{SE}**

\[ K \leftarrow \mathcal{R} \{0, 1\}^k \]
\[ b' \leftarrow \mathcal{R} \mathcal{A}^{Enc} \]
\[ \text{ret } b' \]

\[ Enc(N, M) \]
\[ C \leftarrow \text{enc}(K, N, M) \]
\[ C \leftarrow \mathcal{R} \{0, 1\}^{\lvert C \rvert} \]
\[ \text{ret } C \]

\[ Dec(N, C) \]
\[ \text{Ret } \bot \]

**Nonce-respecting:**

Never repeat N to Enc
Can’t query Dec(N,C) if (N,C) from Enc(N,M)

**Repeat-respecting:**

Never repeat (N,M) to Enc
Can’t query Dec(N,C) if (N,C) from Enc(N,M)

See [Rogaway, Shrimpton 2006]

Simple strengthening of CCA security to also prevent forgeries

\[ \text{CTXT } + \text{ Nonce-resp. Ind}\$ \iff \text{ Nonce-resp. All-in-one Ind}\$ \]

\[ \text{CTXT } + \text{ Repeat-resp. Ind}\$ \iff \text{ Repeat-resp. All-in-one Ind}\$ \]