Side-Channel Attacks
and Chosen Ciphertext Security

Mike Reiter

1

Chosen Ciphertext Attacks

- **Recall that in a chosen ciphertext attack, the adversary is given**
  - An encryption oracle E_K
  - A decryption oracle D_K
  - A test oracle T_K
    - If $c \leftarrow T_K(m_0, m_1)$ then adversary is not permitted to invoke $D_K(c)$
- **Arguably having otherwise unfettered access to D_K is unrealistic, and so variations on this model have been explored**
  - Lunchtime attack: Adversary can query D_K only before querying T_K
  - Side-channel attack: Instead of having access to D_K, adversary is given access to a "side channel" oracle P_K
    - $P_K(c)$ returns $f(D_K(c))$ for a particular function f
- **We will explore a frequently practical side channel in this lecture**

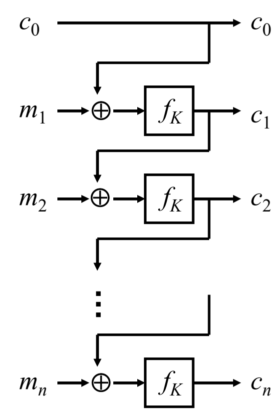
2

Recall CBC Mode Encryption

- **Let f: Keys $\times$ $\{0,1\}^L \rightarrow \{0,1\}^L$ be a pseudorandom *permutation***

<u>Algorithm</u> $E_K(m)$:
let $m_1|\ldots|m_n = m : m_i \in \{0,1\}^L$
$c_0 \leftarrow_R \{0,1\}^L$
for $i = 1\ldots n$ do $c_i \leftarrow f_K(c_{i-1} \oplus m_i)$
return $c_0 \mid c_1 \mid \ldots \mid c_n$

<u>Algorithm</u> $D_K(c)$:
let $c_0 \mid c_1 \mid \ldots \mid c_n = c : c_i \in \{0,1\}^L$
for $i = 1\ldots n$ do $m_i \leftarrow f_K^{-1}(c_i) \oplus c_{i-1}$
return $m_1 \mid \ldots \mid m_n$



- **Above description assumes that length of m is a multiple of L**
  - If not, <u>padding</u> is required

3

Padding

- **A padding function is a function PAD: $\{0,1\}^* \rightarrow (\{0,1\}^L)^+$**
  - Most applications require PAD to be reversible
- **Two types of padding functions**
  - <u>Byte-oriented</u>, where PAD: $(\{0,1\}^8)^+ \rightarrow (\{0,1\}^L)^+$ and $L = 8b$
  - <u>Bit-oriented</u>, where domain of PAD is unrestricted
- **Example: CBCPAD is a byte-oriented padding function**

<u>Algorithm</u> CBCPAD(m):
let $m_1|\ldots|m_n = m : m_i \in \{0,1\}^8$
$p \leftarrow b - (n \bmod b)$
return $m \mid \underbrace{pp\ldots p}_{p \text{ times}}$

- **Padding is "01", "02 02", "03 03 03", "04 04 04 04" …**
  - Denote this by "$p \times p$"

4

Processing Padding

5

- **What if the padding in a ciphertext is not valid?**
  - tear down the session (as in SSL/TLS)?
  - log the error (as in ESP)?
  - return an error message (as in WTLS)?
- **Either way, typically will leak whether the padding was valid**
- **Abstract this as an oracle P_K**

Algorithm $P_K(c)$:
let $c_0 \mid c_1 \mid \ldots \mid c_n = c : c_i \in \{0,1\}^L$
for $i = 1 \ldots n$ do $m_i \leftarrow f_K^{-1}(c_i) \oplus c_{i-1}$
if m_n ends in $p \times p$ for some $p > 0$
return 1
else
return 0

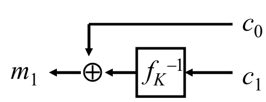
5

Last Byte Decryption

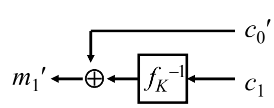
[Vaudenay 2002]

6

- **Consider a two-block ciphertext $c_0 \mid c_1$**
- **We know that decryption is performed as follows**



- **Consider any $c_0' \neq c_0$**



- **Since** $c_0 \oplus m_1 = c_0' \oplus m_1' = f_K^{-1}(c_1)$**, we get** $m_1 = (c_0 \oplus c_0') \oplus m_1'$
  - We know $(c_0 \oplus c_0')$ but not m_1'

6

Last Byte Decryption (cont.)

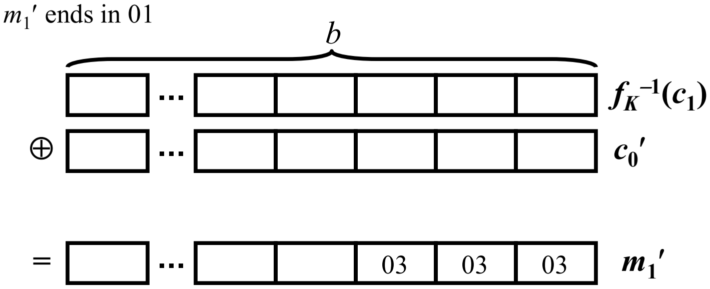
- **However, if we can find c_0' such that $P_K(c_0' \mid c_1) = 1$, then we know that m_1' is correctly padded**

- **Moreover, if c_0' is chosen randomly from $\{0,1\}^L$, then**
  - m_1' ends in 01 with probability $1/2^8$
  - m_1' ends in 02 02 with probability $1/2^{16}$
  - m_1' ends in 03 03 03 with probability $1/2^{24}$
  - ...

- **So, we could just assume that m_1' ends in 01, and would usually be right**
  - If correct, then last byte of m_1 is last byte of $(c_0 \oplus c_0') \oplus 01$

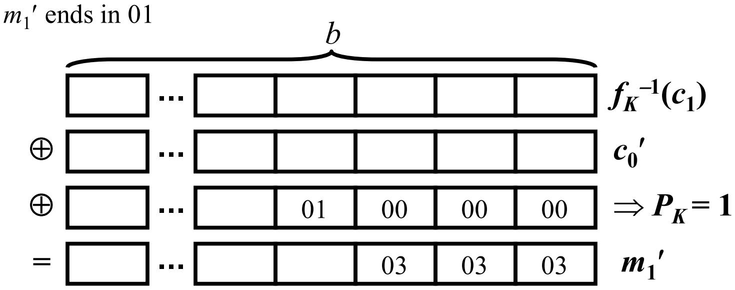
Last Byte Decryption (cont.)

- **To get it right in all cases, start from c_0' where $P_K(c_0' \mid c_1) = 1$ and do the following**
  - If $P_K((c_0' \oplus 01(00)^{b-1}) \mid c_1) = 0$ then m_1' ends in $b \times b$, else
  - If $P_K((c_0' \oplus 01(00)^{b-2}) \mid c_1) = 0$ then m_1' ends in $b-1 \times b-1$, else
  - ...
  - If $P_K((c_0' \oplus 01(00)^{1}) \mid c_1) = 0$ then m_1' ends in 02 02, else
  - m_1' ends in 01



Last Byte Decryption (cont.)

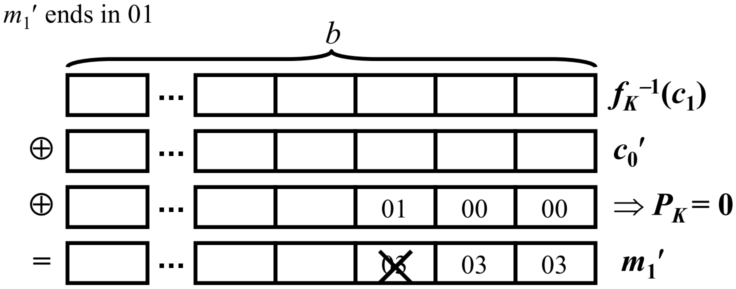
- **To get it right in all cases, start from c_0' where $P_K(c_0' \mid c_1) = 1$ and do the following**
  - If $P_K((c_0' \oplus 01(00)^{b-1}) \mid c_1) = 0$ then m_1' ends in $b \times b$, else
  - If $P_K((c_0' \oplus 01(00)^{b-2}) \mid c_1) = 0$ then m_1' ends in $b-1 \times b-1$, else
  - ...
  - If $P_K((c_0' \oplus 01(00)^{1}) \mid c_1) = 0$ then m_1' ends in 02 02, else
  - m_1' ends in 01



9

Last Byte Decryption (cont.)

- **To get it right in all cases, start from c_0' where $P_K(c_0' \mid c_1) = 1$ and do the following**
  - If $P_K((c_0' \oplus 01(00)^{b-1}) \mid c_1) = 0$ then m_1' ends in $b \times b$, else
  - If $P_K((c_0' \oplus 01(00)^{b-2}) \mid c_1) = 0$ then m_1' ends in $b-1 \times b-1$, else
  - ...
  - If $P_K((c_0' \oplus 01(00)^{1}) \mid c_1) = 0$ then m_1' ends in 02 02, else
  - m_1' ends in 01



10

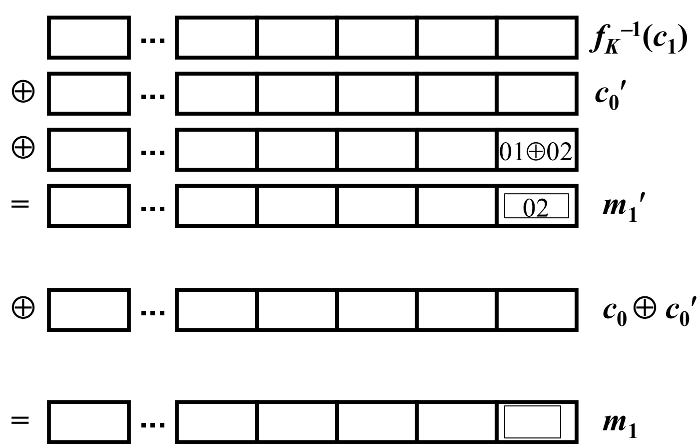
Block Decryption

- Now we can use this to find all of m_1

11

Block Decryption

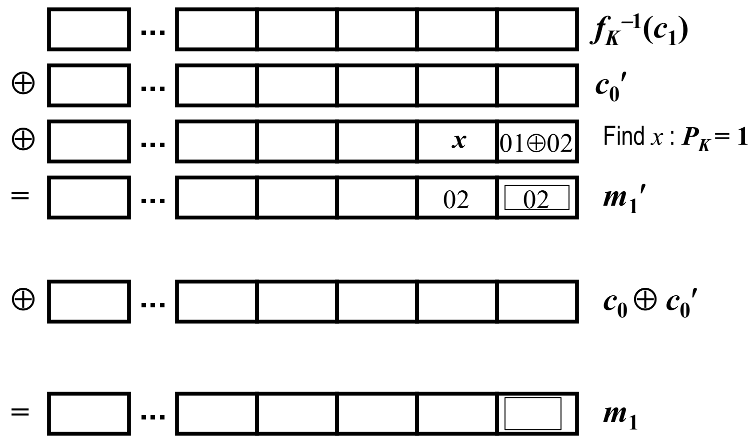
- Now we can use this to find all of m_1



12

Block Decryption

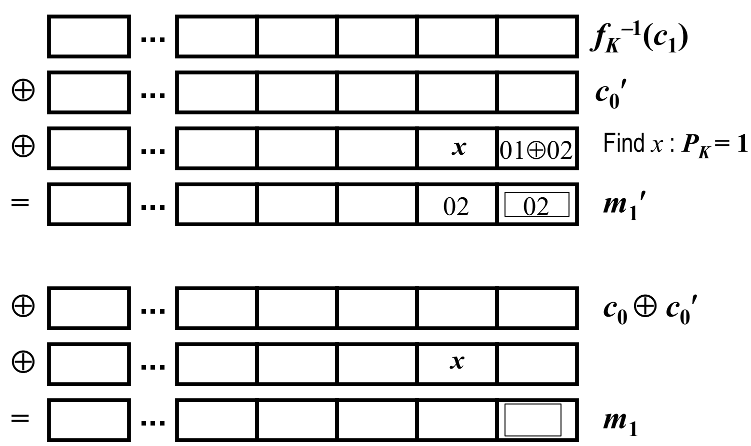
- Now we can use this to find all of m_1



13

Block Decryption

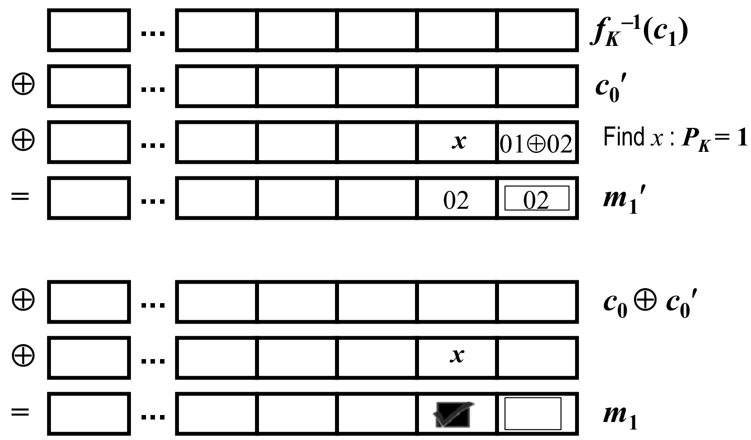
- Now we can use this to find all of m_1



14

Block Decryption

- **Now we can use this to find all of m_1**



15

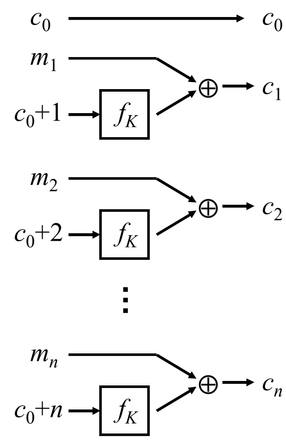
Full Decryption

- **Once we've implemented block decryption, full decryption of multi-block messages is straightforward**
  - Do each block separately
  - Use preceding ciphertext block as its initialization vector
- **Block decryption can be sped up using binary search instead of linear search to find padding length**

16

Other Symmetric Encryption Schemes

- CBC is not the only encryption mode where padding is used
- Recall *counter mode*



Algorithm $E_K(m)$:
 let $m_1|\ldots|m_n = m : m_i \in \mathbf{\{0,1\}}^L$
 let $c_0 \leftarrow_R \{0,1\}^L$
 for $i = 1\ldots n$ do $c_i \leftarrow f_K(c_0+i \bmod 2^L) \oplus m_i$
 return $c_0 \mid c_1 \mid \ldots \mid c_n$

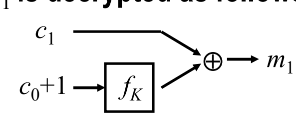
Algorithm $D_K(c)$:
 let $c_0 \mid c_1 \mid \ldots \mid c_n = c : c_i \in \mathbf{\{0,1\}}^L$
 for $i = 1\ldots n$ do $m_i \leftarrow f_K(c_0+i \bmod 2^L) \oplus c_i$
 return $m_1 \mid \ldots \mid m_n$

- Padding here is similarly tricky

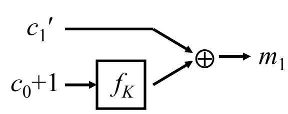
17

Counter Mode Encryption

- Suppose CBCPAD were used with counter mode
- A ciphertext $c_0 \mid c_1$ is decrypted as follows



- For any $c_1' \neq c_1$

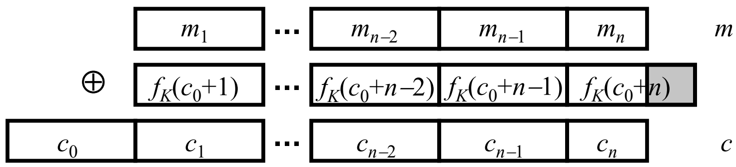


- Since $c_1 \oplus m_1 = c_1' \oplus m_1' = f_K(c_0+1)$, we get $m_1 = (c_1 \oplus c_1') \oplus m_1'$
- Once again, padding oracle enables m_1 to be recovered

18

Counter Mode Encryption (cont.)

- **Note, however, that unlike with CBC, padding is not *necessary* with counter mode encryption**



- **Blue portion can be discarded, rather than padding to utilize it**
  - Advantage: eliminates any padding oracle
  - Disadvantage: exposes exact bit length of plaintext

Other Byte-Oriented Padding Schemes

- **IPSec Encapsulated Security Payload (ESP-PAD)**
  - To pad with $p > 0$ bytes, use 01 02 … p
  - Also vulnerable in the same way
- **Prefix padding**
  - Use CBCPAD but at front of message
  - Still vulnerable to same attack, and requires more state to encrypt
- **Last byte = padding length**
  - Last byte is length of padding; all other padding bytes random
  - Can be used in roughly same way, but to extract only the last byte of each plaintext block

Other Byte-Oriented Padding Schemes (cont.)

- ***XY* padding**
  - Let X and Y be two distinct public constants
  - Pad with X followed by as many Y's as needed (possibly 0)
  - Also vulnerable in (roughly) the same way
- **Any-pair padding**
  - Like XY padding, but X and Y are chosen randomly per message, and Y must be appended at least once
  - All ciphertexts have valid padding, except those with all plaintext bytes being equal
  - Eliciting a 0 from the oracle requires expected 2^{L-9} queries if plaintexts are random (which they're not)

21

Other Byte-Oriented Padding Schemes (cont.)

- **Any-tail padding**
  - Message padded with any random Y (at least once) that is distinct from the last byte X of the plaintext
  - All ciphertexts have valid padding
    - Padding oracle is eliminated
  - Has an obvious bit-oriented analog
- **Padding followed by integrity check**
  - Message is padded, and then hash(message|padding) is appended before encryption
  - Important for hash to be performed *after* padding, and checked before padding is checked on receiver side
  - Virtually eliminates padding oracle (but has other weaknesses)

22

Aborts

- **Some protocols (notably SSL/TLS) abort if they encounter a padding error**
  - If ciphertext is not authenticated, this is denial-of-service vulnerability
  - If ciphertext is authenticated, then padding oracle is unavailable
- **Aborts limit the attacker to one guess**
- **If the receiver does not abort, then attacker learns last byte of plaintext for whatever ciphertext he submitted**
  - Succeeds with probability $\approx 1/2^8$

Padding Oracles in Public Key Systems

[Bleichenbacher 1998]

- **Public key systems are equally vulnerable to attacks using padding oracles**
- **Recall RSA cryptosystem**
  - Public key $K = \langle e, N \rangle$, where $N = pq$ for primes p, q
  - Private key $K^{-1} = \langle d, N \rangle$, where $ed \equiv 1 \bmod (p-1)(q-1)$
  - $E_K(m) = (\text{pad}(m))^e \bmod N$
  - $D_{K^{-1}}(c) = \text{pad}^{-1}(c^d \bmod N)$

PKCS #1 v1.5 Padding for Encryption

- **If $|N| = k$ bytes, then** $256^{k-1} < N < 256^k$
- **PKCS #1 (v1.5) padding for encryption is correct if**
  - 1st byte is 00
  - 2nd byte is 02
  - next 8 bytes different from 00
  - at least one more 00 byte

most significant byte ↓ … ↓ least significant byte

00	02	padding string	00	message

padding string: ← at least 8 bytes →

← k bytes →

25

Properties of PKCS #1 v1.5 Padding

- **Probability Pr(PKCS) that a random message is correctly padded is**

$$0.18 \cdot 2^{-16} < \Pr(\text{PKCS}) < 0.97 \cdot 2^{-8}$$

- **1/Pr(PKCS) < 360,000**
  - PKCS conforming messages can be found by trial and error
- **Given a target ciphertext $c = m^e \bmod N$, attacker can submit $c_i = c(s_i)^e \bmod N$ to the padding oracle**
  - If c_i is PKCS conforming, then $2 \cdot 256^{k-2} \leq ms_i \bmod N < 3 \cdot 256^{k-2}$
- **This fact can be leveraged to decrypt c**

26

Cost of Attack

- **Number of queries needed**
  - Pr(PKCS) = probability that a random message is PKCS conforming
    - $0.18 \cdot 2^{-16} < \Pr(\text{PKCS}) < 0.97 \cdot 2^{-8}$
  - Pr(PKCS|A) = probability that a message with leading bytes 00 and 02 is PKCS conforming
    - $0.18 < \Pr(\text{PKCS}|A) < 0.97$
  - Number of oracle queries is $\approx 3/\Pr(\text{PKCS}) + 16k/\Pr(\text{PKCS}|A)$

- **For example, if N is 1024 bits then roughly 1,000,000 queries are needed**

Length-Revealing Oracles

- **Padding oracles are not the only side channels**
- **Consider a length-revealing oracle**
  - Given ciphertext input, returns the length of the plaintext (with padding stripped)
- **May result from link encryption**
  - Outgoing link reveals length of incoming plaintext

Incoming encrypted with CBC →

→ Outgoing encrypted with Counter Mode (no padding)

  - This can be used to defeat even the previous "good" padding schemes
    - (Work some examples)

Chosen Ciphertext Security

- **These various side-channel attacks motivate the need for chosen ciphertext security**
- **Any adversary that can succeed using a side-channel attack can succeed using a chosen-ciphertext attack**
  - simply uses the decryption oracle to implement the side channel
- **Conversely, encryption that is invulnerable to chosen ciphertext attacks is also invulnerable to side channel attacks (based on the output from the decrypting party)**

29

Recall a Chosen Ciphertext Attack

- **The adversary is given three oracles**
  - An encryption oracle E_K
  - A test oracle $T_K(m_0, m_1)$ that can be called only once

    Oracle $T_K(m_0, m_1)$:
    if $|m_1| \neq |m_2|$ then return $\perp$
    $b \leftarrow_R \{0,1\}$
    return $E_K(m_b)$

  - A decryption oracle D_K
- **The adversary must guess whether $b = 0$ or $b = 1$, but if**

$$c \leftarrow T_K(m_0, m_1)$$

then adversary cannot query $D_K(c)$

30

Definition of CCA Security

- A CCA-secure encryption scheme is a triple

$$\langle \mathbf{Gen}, E, D \rangle$$

such that for every PPT A there is a negligible ν_A where

$$\Pr[\, A^{E_K, D_K, T_K} = 0 : b \leftarrow 0 \,] - \Pr[\, A^{E_K, D_K, T_K} = 0 : b \leftarrow 1 \,] \;\le\; \nu_A(\lambda)$$

for all sufficiently large λ, where

- the probabilities are taken over $K \leftarrow \mathbf{Gen}(1^\lambda)$
- A^{E_K, D_K, T_K} is not permitted to query $D_K(c)$ if $c \leftarrow T_K(m_0, m_1)$