Encryption Attack Details, Hash Functions
CMSC 23200/33250, Autumn 2018, Lecture 5

David Cash
University of Chicago
Plan

1. Padding Oracle Attack Details
2. Introduction to Cryptographic Hash Functions
3. Pitfalls with Hash Functions
4. Begin Public-Key Encryption
Assignment 1 is Coming Today

Updated due-date will be announced
AES-CBC with Padding

**AES-CBC Encryption:**

- IV
- $M_1$
- $M_2$
- $IV$
- $C_1$
- $C_2$

**AES-CBC Decryption:**

- IV
- $C_1$
- $C_2$
- $IV$
- $M_1$
- $M_2$

AES$_K()$ and AES$^{-1}_K()$ indicate encryption and decryption functions, respectively.
PKCS7 Padding

- Need to pad a byte string up to a multiple of 16 bytes
- First look at how many bytes are missing. Here, need 10 bytes
- Fill missing k bytes with value k (k = 10 = 0x0A in example)

- If data is already a multiple of 16 bytes long, add an entire block of 0x10 bytes

Can’t leave data unchanged;
Bytes might be interpreted as padding.

- Un-padding is easy
Invalid Padding

**Fact:** Not every byte string is a “valid padding”. Some strings have to be handled as “malformed”

Invalid. Why?

Valid. Why?

Valid! All strings that end in 0x01 are valid.
Padding Oracle Attacks

1. $M' \leftarrow \text{CBC-Decrypt}_{K1}(\text{IV}, C_1, C_2)$
2. If padding format wrong:
   - Output PADDING_ERROR
3. If tag $T$ wrong:
   - Output REJECT.
4. Output $M'$

Real-world attacks against:
- TLS (2003)
- IPSec (2007, 2010)
- Ruby on Rails (2010)
- ASP.NET (2010)
- SecurID Auth Tokens (2012)
- Steam Client (2016)

Now: How to find one byte of plaintext.

System (e.g. webserver)

$K$
Attack Setting

- First two blocks of long valid ciphertext are pictured.

Have this, plus the padding oracle.
Initial goal: Learn last byte of a block

- First observation: If we truncate the long ciphertext down to these blocks, what will padding oracle say?
  - Padding_Error!

- What can we infer from the padding error response?
  - Last plaintext byte of that block could not have been 0x01.
Initial goal: Learn last byte of a block.

- What will padding oracle say? 
  -(still) \texttt{Padding\_Error}! 

After decryption, this byte becomes 0x10.

Consider changing this byte to 0x65, then submitting ciphertext to oracle.
Initial goal: Learn last byte of a block.

- What will padding oracle say?
  - Padding is valid!
  - Infer value of last byte: $0x10 \oplus 0x01 = 0x11$

After decryption, this byte becomes $0x01$.

Consider instead changing this byte to $0x10$, then submitting ciphertext to oracle.
Fact: If last byte of plaintext is $x$, and we change last byte of prior ciphertext block to $x \oplus 0x01$, then padding will be valid. Changing that byte to anything else will usually result in invalid padding.

Q: How do we know what to set that byte to?  
A: Just try all 256 until one works.
Recovering more plaintext bytes

- Assume we know last byte is 0x11.
- Eventually get valid padding with 0x02 bytes!
- Guess plaintext byte as before.

Eventually this becomes 0x02 too.

Becomes 0x02

Change to 0x11 ⊕ 0x02

Now start changing this byte.
Assignment 1: Full plaintext recovery

- Extend idea using cases with larger padding to recover entire block
- Nothing special about the second block; We can take any block and apply attack.
- Some possibility for false positives (be careful).
Lessons

- Always use Authenticated Encryption, with properly implemented integrity checks.
- If a library provides Authenticated Encryption, always try to use it.
  - AES-GCM is a good choice
- If not, then use AES-CTR + a good MAC like HMAC.
  - It’s easy and compact
  - Components are very common
  - Don’t implement AES-GCM, especially if you don’t know what a Galois Field is. (But don’t even if you do)
Another avenue for attacks: Compress-then-Encrypt

Compress()
Compression Attack Setting: Browser Data

Real-world attacks against:
- TLS (2012)
- HTTPS (2013)

Adversary can see ciphertext length.

More overlap with secret means better compression, i.e. shorter ciphertext.

Assignment 1: Seeing several ciphertexts enables full plaintext recovery.
**Definition:** A hash function is a deterministic function $H$ that reduces arbitrary strings to fixed-length outputs.

- **MD5:** $m = 128$ bits
- **SHA-1:** $m = 160$ bits
- **SHA-256:** $m = 256$ bits
- **SHA-512:** $m = 512$ bits
- **SHA-3:** $m \geq 224$ bits

Some security goals:
- collision resistance: can’t find $M \neq M'$ such that $H(M) = H(M')$
- preimage resistance: given $H(M)$, can’t find $M$
- second-preimage resistance: given $H(M)$, can’t find $M'$ s.t. $H(M') = H(M)$

Note: Very different from hashes used in data structures!
Hash Functions are not MACs

Both map long inputs to short outputs… But a hash function does not take a key.

Intuition: a MAC is like a hash function, that only the holders of key can evaluate.
Hash Function Security History

- Can always find a collision in $2^{m/2}$ time ($\ll 2^m$ time). “Birthday Attack”
- MD5 (1992) was broken in 2004 - can now find collisions very quickly.
- SHA-1 (1995) was broken in 2017 - A big computer can find collisions
- SHA-256/SHA-512 (2001) are not broken
- SHA-3 (2015) is new and not broken

```
MD5(d131dd02c5e6ec4693d9a0698aff95c 2fcab58712467eab4004583eb8fb7f89
  55ad340609f4b30283e488832571415a 085125e8f7cdc99fd91dbdf280373c5b
d8823e3156348f5bae6dacd436c919c6 dd53e2b487da03fd02396306d248cda0
e99f33420f577ee8ce54b67080a80d1e c69821bcb6a8839396f9652b6ff72a70)
```

= MD5(d131dd02c5e6ec4693d9a0698aff95c 2fcab50712467eab4004583eb8fb7f89
  55ad340609f4b30283e488832571415a 085125e8f7cdc99fd91dbdf280373c5b
d8823e3156348f5bae6dacd436c919c6 dd53e23487da03fd02396306d248cda0
e99f33420f577ee8ce54b67080280d1e c69821bcb6a8839396f9652b6ff72a70)

Xiaoyun Wang (Tsinghua University), 2004
  - Broken with clever techniques
  - Compare to DES (broken b/c key too short)
MACs from Hash Functions

Goal: Build a secure MAC out of a good hash function.

Common construction: \( \text{MAC}(K, M) = H(K \| M) \)

- Totally insecure if \( H = \text{MD5}, \text{SHA1}, \text{SHA-256}, \text{SHA-512} \)
- Is secure with \( \text{SHA-3} \)

Upshot: Use HMAC and avoid various issues.

Later: Hash functions and certificates
The End