Crypto 2: Collision Boogaloo

Deepak Kumar

Educational Goals

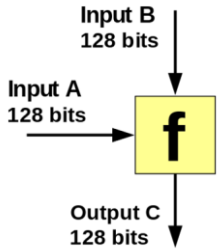
- Clearly understand what you need to do for each part of the checkpoint
- Learn the Merkle-Damgard construction and why it enables length extension attacks
- Understand CBC mode intricately and how the padding oracle attack works
- Learn what a product tree is and how to implement it for factoring weak RSA moduli
- Understand the certificate collision algorithm

Length Extension Attacks: Merkle-Damgard

- Recall: Hash function maps arbitrary input to fixed output space, should be *collision resistant*
- MD5, SHA1, SHA2 all use Merkle-Damgard to build a collision resistant hash function
- Merkle-Damgard leverages chains of *one-way compression functions* to build a *collision-resistant* hash function

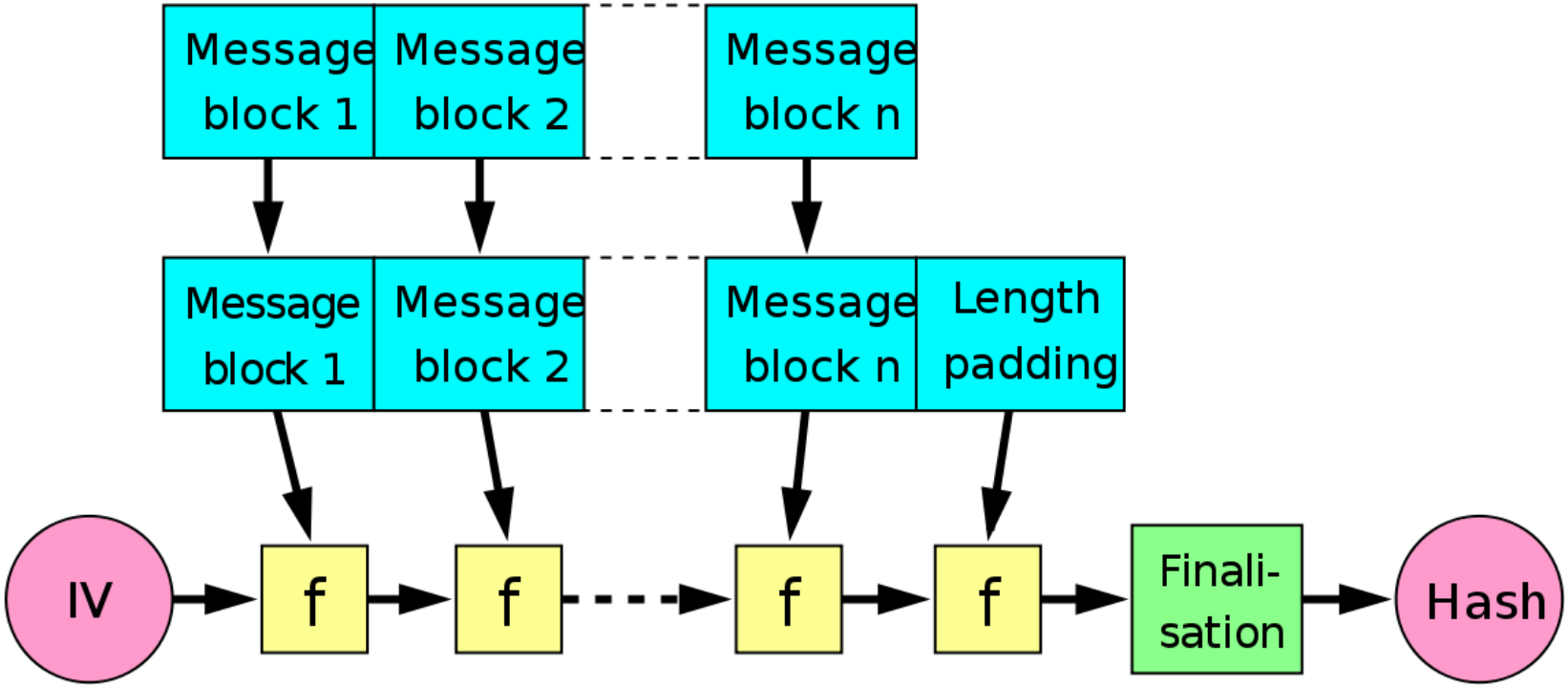
Length Extension Attacks: Merkle-Damgard

- Recall: Hash function maps arbitrary input to fixed output space, should be *collision resistant*
- MD5, SHA1, SHA2 all use Merkle-Damgard to build a collision resistant hash function
- Merkle-Damgard leverages chains of *one-way compression functions* to build a *collision-resistant* hash function



Example compression function

Length Extension Attacks: Merkle-Damgard



Length Extension Attacks: Merkle-Damgard

- Merkle-Damgard is vulnerable to *length extension*
  - Given H(X) of some unknown X, it's easy to know H(Pad(X)||Y)
- Demo

Length Extension Attacks: What you need to do

- You will need to launch a length extension attack on a URL we provide
- The website uses an MD5 hash of the input in the URL as an “authenticator” for what commands are executed

http://ece422.org/project3/api?token=b301afea7dd96db3066e631741446ca1&user=admin& command1=ListFiles&command2=NoOp

- You don’t know the password, but you want to add a command and update the hash accordinly

MD5 Collisions: Background

- In 2019, we can generate MD5 collisions fairly quickly
  - Use `fastcoll`, a program we link to in the MP spec
- What can you do if two programs share the same hash?

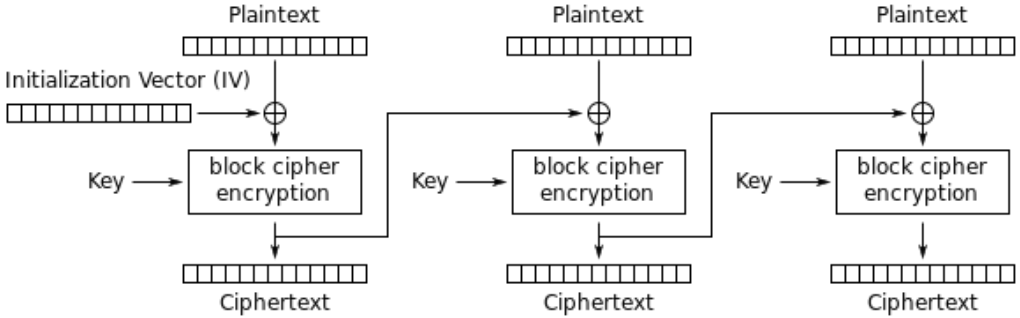
MD5 Collisions: What you need to do

- Write two python programs, good.py and evil.py, that have the same MD5 hash but are fundamentally different programs
- Prerequisite: You can use fastcoll to generate two files that have the same hash and the same prefix
- **Hint**: Think about how you might use the collision to create output that is different in both files.

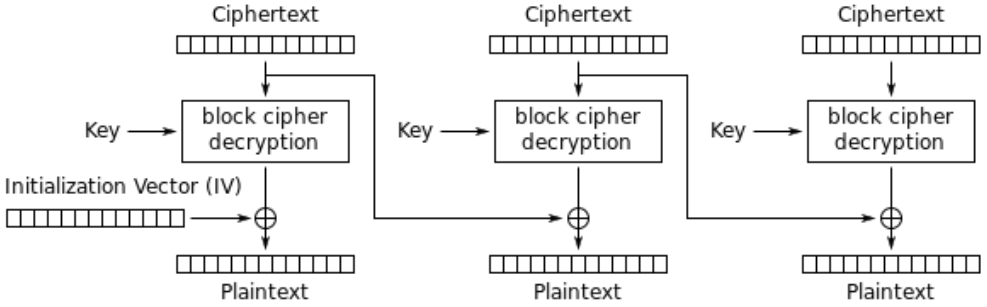
Padding Oracle Attack

- Leverages *knowledge of padding correctness* to entirely learn the plaintext of a message without learning the secret key
- Works on CBC mode with any block cipher

Padding Oracle Attack: Revisiting CBC



Cipher Block Chaining (CBC) mode encryption



Cipher Block Chaining (CBC) mode decryption

Padding Oracle Attack: What you need to do

- Launch a padding oracle attack on some ciphertext we provide to you
- We have a “padding oracle” running at

http://cs461-mp3.sprai.org:8081/

- You can run a local server using the code we provide you to test that your script works

Padding Oracle Attack: How the attack works

- If we learn if the padding is incorrect or correct on decryption of a chosen ciphertext, we can *learn the entirety of the plaintext*
- We can use some tricks
  - XOR is reversible
    - x XOR y = z; z XOR y = x
  - XOR with yourself is 0
  - We have control of the entirety of the ciphertext and the IV
- We can change one byte of the ciphertext at a time to learn the entirety of the plaintext
  - Changes the “brute force” attack from O(2^n) to O(n) for n-bits

Padding Oracle Attack: How the attack works

- Demo

Weak RSA Key Generation – Textbook RSA

Choose two large primes, *p* and *q* at random

$N = p * q$

Choose *e* such that $GCD(e, (p - 1) * (q - 1)) = 1$, *coprime*

Find d such that $e * d \equiv 1 \bmod ((p - 1) * (q - 1))$, *d is modular multiplicative inverse of e for mod N*

Public Key: (e, N)

Private Key: (d, N)

Weak RSA Key Generation – Textbook RSA

Choose two large primes, *p* and *q* at random

N = p * q

What happens when p or q are shared between distinct moduli?

Weak RSA Key Generation: What you need to do

- Given many moduli that share factors for RSA moduli generation, we can efficiently factor all of them
- In this part of the MP, you have 10,000 RSA moduli
  - Public exponent = 65537
  - One of these moduli was used to encrypt a message to you
  - You need to private key, which means you need to finds ps and qs
  - You *could* take the GCD of each pairwise moduli, but you can't do it in time
- Use the methods from Heninger et. al in "Mining Your Ps and Qs" to factor all the moduli and find the corresponding private key
  - Product Trees!
- We give you pbp.py, a script that does the decryption for you given a created RSA private key

Weak RSA Key Generation

- Demo

Colliding Certificates: Background

- *Certificate Authorities* (CAs) issue TLS certificates – these verify the identity of remote servers on the web
- Your browser has a set of known CAs that it trusts to issue certificates – this is called the *root store*
- End-entity certificates are called *leaf* certificates
- Leaf certificates are *signed* by CAs with their private key – this way browsers can verify that the certificates are legit
- In some cases (like this MP), a CA will sign an MD5 *hash* of the certificate data – but as we know, MD5 is prone to collisions!

Colliding Certificates: What you need to do

- Generate two certificates, certA and certB, that have different fields but share a signature from a CA
- We link you to the algorithm you will implement in Lenstra et. al's paper
- You will leverage the starter code provided in mp3-certbuilder.py and use fastcoll to generate MD5 collisions
- **Note:** Generating collisions of the right form can sometimes take a while (order of 15 minutes) – make sure you're clear on when you're generating collisions and why