Mason Competitive Cyber

AES Padding Oracle Attacks

Upcoming Competitions

- DamCTF (#damctf2020)
  - This weekend, online, prizes offered
  - Starts 8PM EST Friday
- MetaCTF-DTCC (#meta-dtcc2020)
  - 6 hour CTF this Saturday (10 October)
  - **$1000/member** for the top **3** teams, team size of 4
- Hivestorm (Teams set)
  - 17 October
- MetaCTF CyberGames 2020 (#metactf-cybergames-2020)
  - 24 October
- National CPTC (invite only)
  - 31 October
- USCYBERCOM Hack The Building (no channel yet, some kind of hybrid GMU/UVA team playing)
  - 16-19 November
- C2C CTF 2020 (signups closed, but several people are playing)
  - 6 December

Security news

- No BSides DC this year, so no Pros vs Joes 2020
  - :(
- Zerologon still a thing
- Apple’s TPM is omegaborked
  - https://threatpost.com/apple-t2-flaw-macs/159866/
  - Tl; Dr: Researcher manages to find unpatchable flaw in Mac TPMs that enables, in the worst case, an evil cable attack installing a kernel-level rootkit

ECE 476 in one slide

- Cryptography (Crypt/Crypto (not currency)) is the science of mathematical trust
  - Obscuring messages in a way such that you *trust* that nobody other than the intended recipient can read them
  - Verifying data in a way such that you *trust* that you can tell whether the data has changed or not
- Basically, just a really useful study of mathematical primitives to keep stuff secret, trusted, and secure
- Definitions
  - Plaintext: Human-readable text which can be turned into useful information with a set of eyeballs and an average brain
  - Ciphertext: encrypted text which should be very difficult to turn into useful information, even with thousands of cores chipping away at it
  - Encryption: The process by which plaintext becomes ciphertext
  - Decryption: The process by which ciphertext becomes plaintext

ECE 476 in ~~one~~ two slides

- Moar definitions
  - Key: Information which is kept secret, and is only known to trusted parties
  - IV: Information to set up a cipher (not always needed!). Best to keep this secret, too.
- The exclusive-or (XOR or $\oplus$) function
  - Is a binary operator
    - Binary operator in both the mathematical and the computer science definition
  - Basically, if only one bit is true, then the output is true, else the output is false

An awful lot of modern crypto relies on XOR, as does the attack we're going through today

Input 1	Input 2	Output
1	1	0
1	0	1
0	1	1
0	0	0

The actual talk starts here

- AES
  - Advanced Encryption Standard
  - Modern block cipher (by modern, I mean older than some of the people listening to this talk)
  - If you want to impress someone, call it Rijndael because that's the actual name of the cipher it's a subset of

- AES is a Symmetric Block Cipher
  - As opposed to a stream cipher

What's the difference?

- In a nutshell: Block ciphers operate on words (fixed length groups of bits), stream ciphers operate on one bit at a time
- If you want to really get into the weeds about it:
  - A basic block cipher is made up of two basic components: a key, and some function to use that key to scramble the plaintext in such a way that only someone with the key can get it back out. Block ciphers have a fixed "block size", or amount of data that they work on. In AES, this block size is 128 bits (16 bytes).

  - A basic stream cipher *also* has two components: a key and a keystream generator. The key is fed into the keystream generator, which produces an infinitely* long keystream which is mixed with the plaintext, usually using XOR

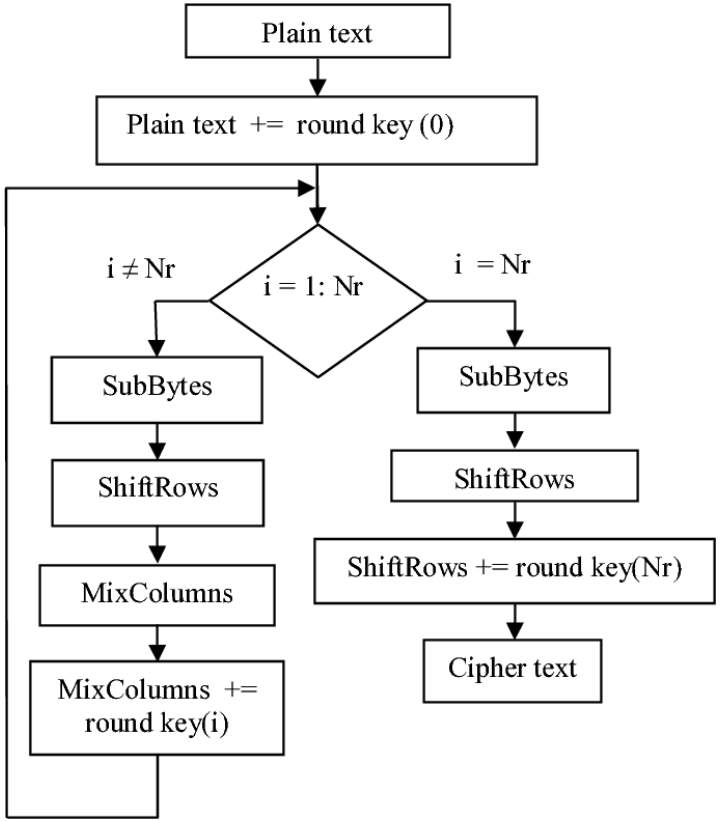
*yes, not actually infinite, but good keystream generators go for a VERY long time before they repeat themselves

Ok, so tell me about AES now

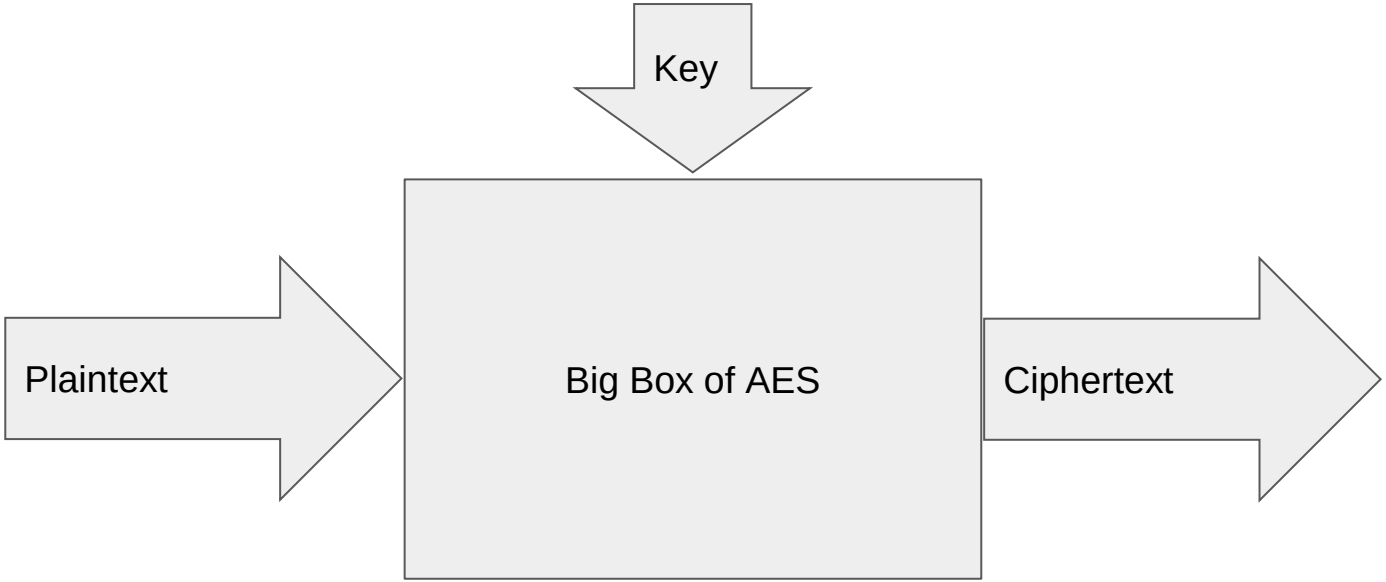
I’m so glad you asked!

AES is at its core the Rijndael round function, which itself consists of five rounds, namely: SubBytes, ShiftRows, MixColumns, and AddRoundKeys. This is augmented by the use of a key expansion algorithm to provide further spreading of entropy within the cipher state. The round function is meant to be repeated several times, either 10, 12, or 14 times, depending on the key size that you feed into AES. AES round operations are made more efficient by operating on a mathematical set called a finite Galois field, which allows for easy translation between mathematical constructs and actual discrete electronic hardware. If you care, AES operates on the GF(2^8) field, AKA GF(256). The SubBytes step of the round is imo the most interesting, because every implementation of AES uses the same lookup table to substitute one byte for another, regardless of the key. So that lookup table’s gotta be some hot shit, right? And it absolutely is, it was designed to be optimized AF while also combatting both linear *and* the more advanced differential cryptanalysis by minimizing the correlation between not just the input and output bits, but by minimizing the correlation between linear transforms OF the input and output bits. In addition to this awesome hardening, the Rijndael S-box also self-optimizes by breaking itself down from 256-value substitution table to a function of two steps, the first of which is taking the multiplicative inverse of the operand byte in the GF(256) finite field and then passing the resulting inverse through an affine transformation which is easy to implement in hardware. The inverse S-Box, which is used in decryption, can be generated on the fly in exactly the same way, just in reverse: an inverse affine transform followed by calculating the multiplicative inverse over GF(256) And that’s just the SubBytes step, there’s FOUR OTHER GLORIOUS STEPS to a round function, and the round function is run up to FOURTEEN times! We haven’t even started talking about the key expansion which is performed, which itself is pretty awesome, carefully optimized and really cleverly hardened, but again imo SubBytes and the Rijndael lookup table are by FAR the coolest part of an AES round. AES is, on a whole, a really sick innovation on the concept of Substitution/Permutation ciphers, and built on pretty much every part of the basics of cryptography.

JK, here's a diagram



Here’s a more realistic diagram



Cipher Modes of Operation

- Ok, so we can encrypt stuff now, am I ready to use my **ELITE** VPN to tunnel my traffic to ~~evade nation state surveillance groups~~ ~~perform journalism from inside a repressive regime~~ ~~secure trade secrets as they pass over international borders~~ make netflix think I'm in Switzerland???

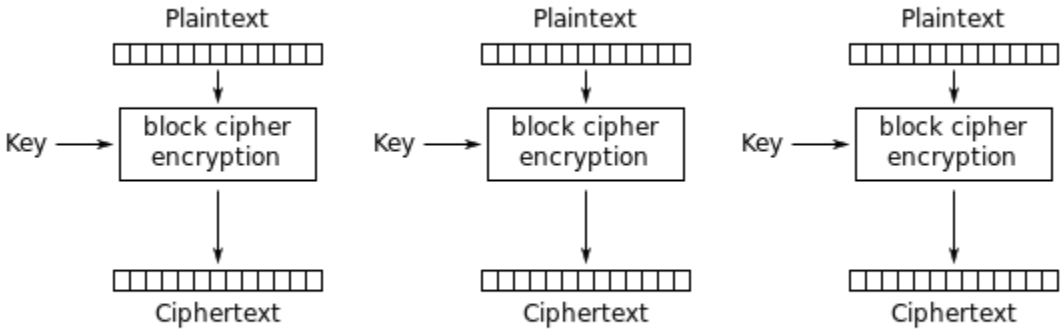
- Don't go out to the internet just yet! You're doing something bad and you don't even know it!

- Block ciphers can be used in several different ways, AKA "Modes of Operation"

- Just encrypting the stuff and sending it over the wire is called "Electronic Code Book" or ECB, and it's BAD

But I *like* ECB

- “It’s simple and easy to understand!”
- Nope, not allowed
- Why?
- Traffic-based attacks
  - Replays, replacements, and rogue comms (oh my)



Electronic Codebook (ECB) mode encryption

Attacking an ECB System

- Remember: AES is a *block* cipher, operating on 16 byte *blocks* of data
- Scenario:
  - You have a really predictable habit of buying Overwatch loot boxes. Like, to the second, every day at 4:30 AM, you buy 50 loot boxes, and frankly you disgust me.
  - I think that *I* could spend your money much better than you could
  - Let's say your payment information goes to PayPal in a form like this, all encrypted with the very *finest* AES-256 Military-Grade Encryption:

16 byte sender (you)	16 byte receiver (Blizzard)	32 bit amount ($)

  - I somehow manage to establish a position on the network where your encrypted payment request flows through me, and I can modify it however I like
  - So now *I* buy some loot boxes, but only 2 though because I'm not an animal. I take the first encrypted block off the network, which I know contains my payment ID:

16 byte sender (Me)	16 byte receiver (Blizzard)	32 bit amount ($)

The actual attack part

- I then my encrypted sender ID and the next morning at 4:30, when you go to spend another $40 on loot boxes, I do this:

16 byte sender (you)	16 byte receiver (Me)	32 bit amount ($)

  I replaced the intended receiver with my encrypted receiver ID, so now the money goes to me. I put it into my 401K and continue to judge your poor spending habits, and you spend 36 hours on hold with Blizzard technical support and never get your money back.

- That doesn't seem like a fun scenario for you, right?

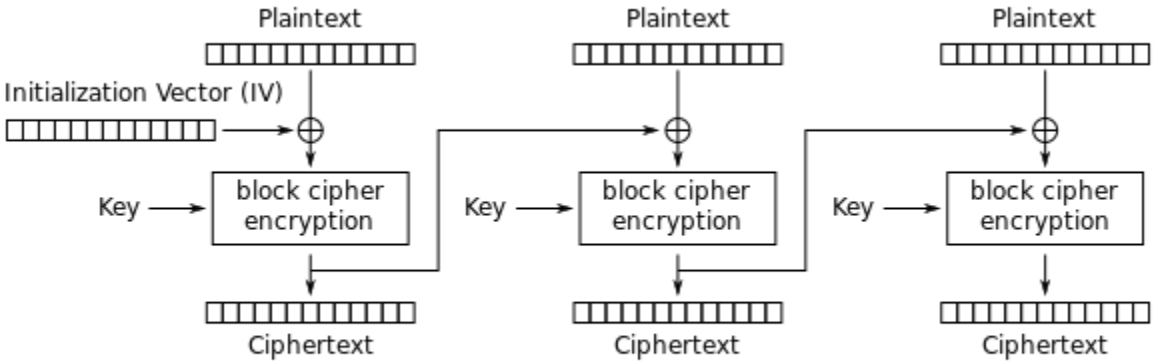
- Well, luckily the cryptographic community agrees, and have come up with several different modes of operation

Cipher Modes of Operation

- MANY to choose from
- OFB, CBC, CFB, CTR, GCM, OMGWTFBBQ

- Today, we're going to be focusing on one mode in particular: CBC, or Cipher Block Chaining (no relation to blockchain, get your mind out of the buzzword gutter)

- This is where IVs come into play!



Cipher Block Chaining (CBC) mode encryption

CBC vs ECB

- In CBC, changes cascade through blocks
- So in the last attack, decryption would fail because we totally changed one block, and no payment would happen
- **BUT**
- CBC isn't a foolproof fix, and actually is pretty inferior to other modes of operation too
  - SSL3
- They say you should never roll your own crypto, and attacks like the one we're about to go over is why: it's VERY easy to accidentally find yourself in a system configuration where the world's best cryptographic algorithm to date (feel free to @ me) is rendered totally ineffective
- Enter: The Padding Oracle

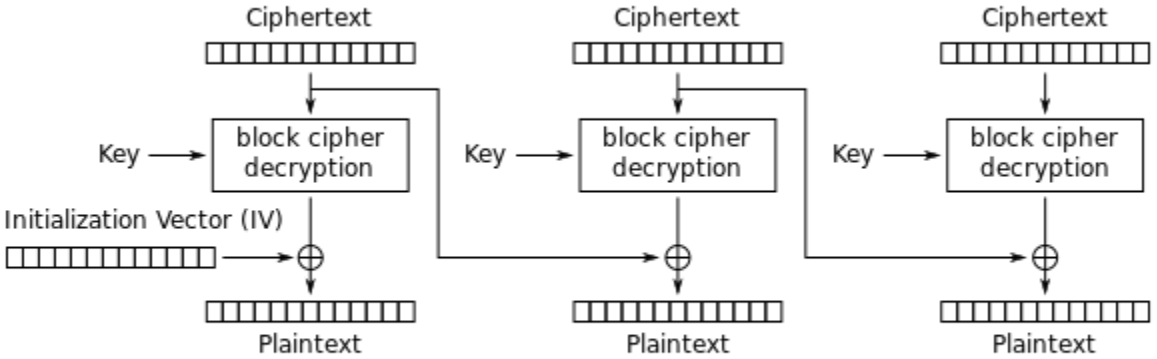
The What?

- The Padding Oracle!

- AES is a block cipher (honestly, add this to the drinking game)
- Block ciphers operate on blocks that are EXACTLY the expected size. No shorter, no longer
- So if a block is too short, it needs to be padded before we can actually send it through the algorithm
- In AES, padding is guaranteed. Even if you have a message that fits exactly within the block size, there's always a whole extra block of padding trailing behind it
- The padding consists of bytes that are the integer value of the amount of bytes of padding that were added
    - Say that five times fast. This scheme is called **PKCS#7 padding** if you want to google it.
- Basically:
    - If you needed to add one byte of padding, you pad with [0x01]
    - If you needed to add 15 bytes of padding, you pad with [0x0F,0x0F,0x0F,0x0F,0x0F,0x0F,0x0F,0x0F,0x0F,0x0F,0x0F,0x0F,0x0F,0x0F,0x0F]
    - If you needed to add 4 bytes of padding, you pad with [0x04,0x04,0x04,0x04]
    - And so on

Ok, but what's the Oracle?

- In cryptography (particularly cryptanalysis), an Oracle is a service that provides information
- This could be intentional or unintentional
- A Padding Oracle will tell you whether your padding was valid or invalid

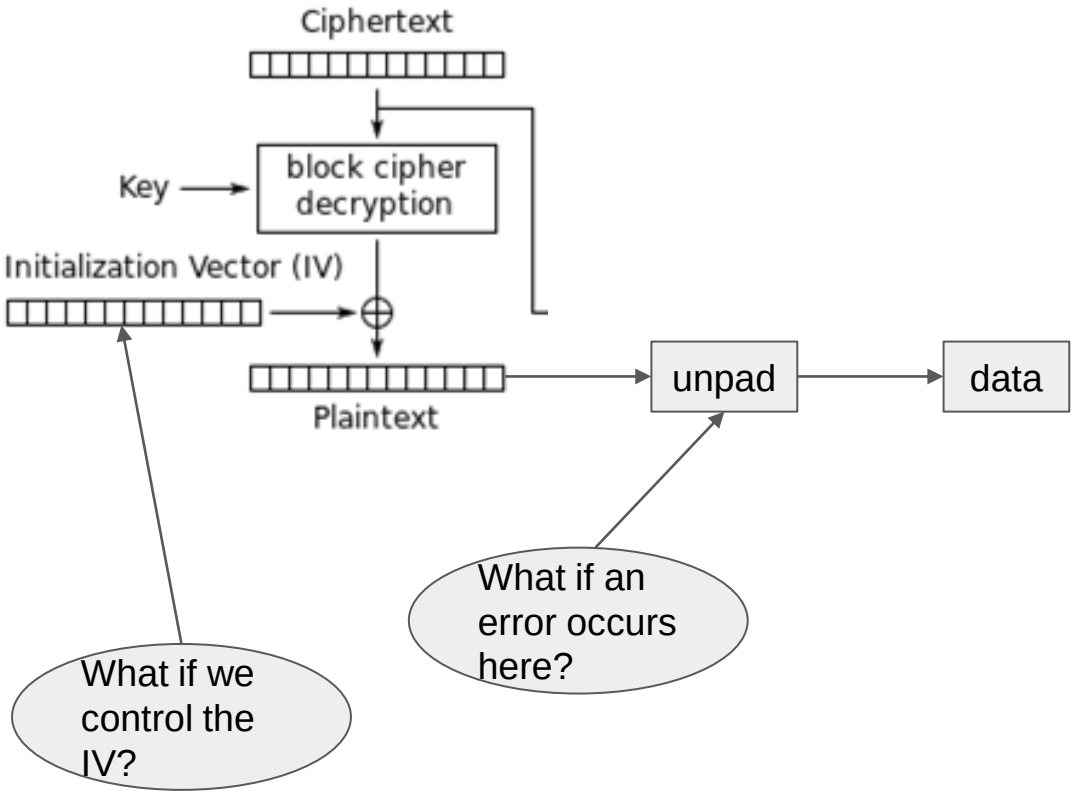


Cipher Block Chaining (CBC) mode decryption

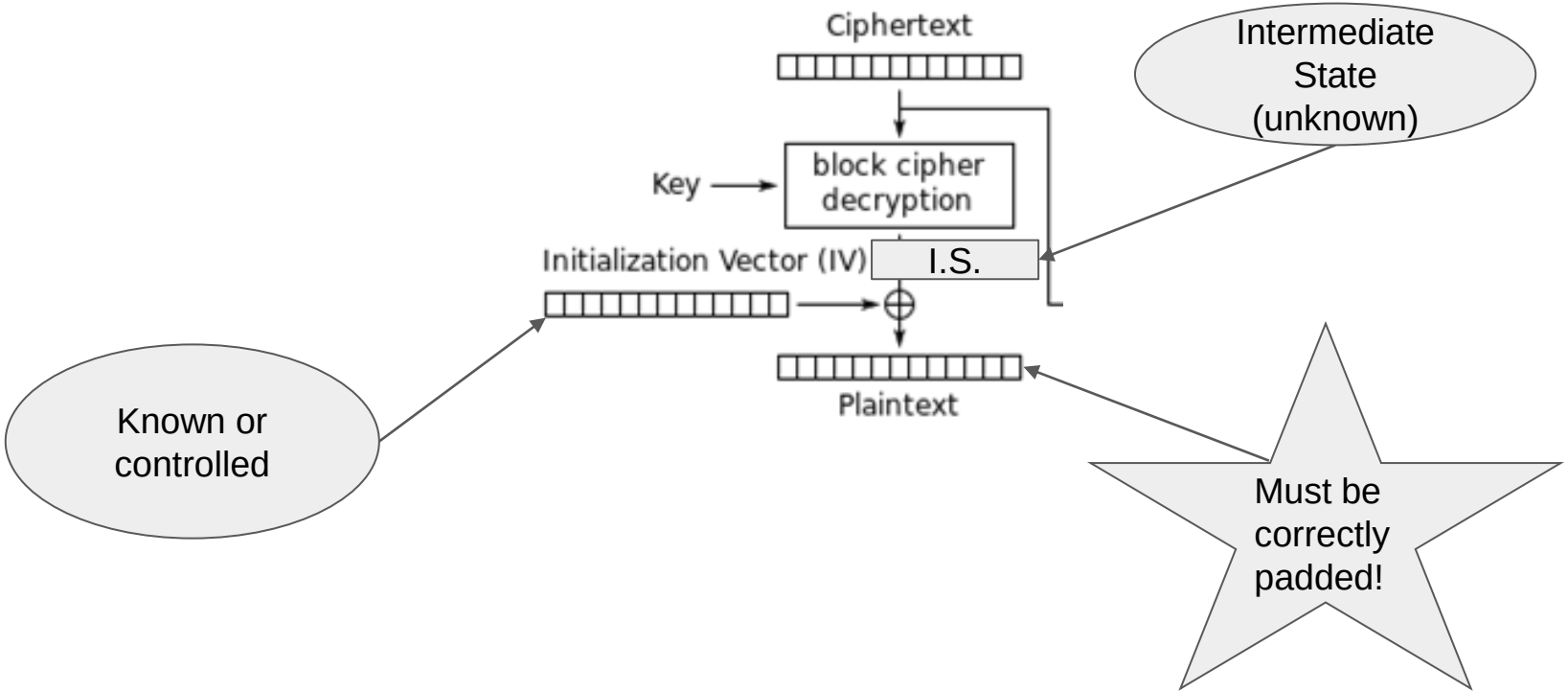
Criteria for a Padding Oracle atk

- A service which decrypts the thing you send it
  - The service doesn’t send the decrypted result back to you (maybe it stores it locally or something)
  - The service reports errors in decryption if any occur
- Some ciphertext encrypted with the same key used by the decryption service
- In the ideal case, the IV used by the service to do encryption

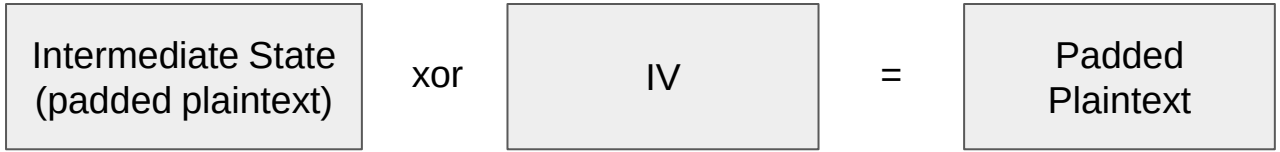
The actual attack, for real



The actual attack, for real

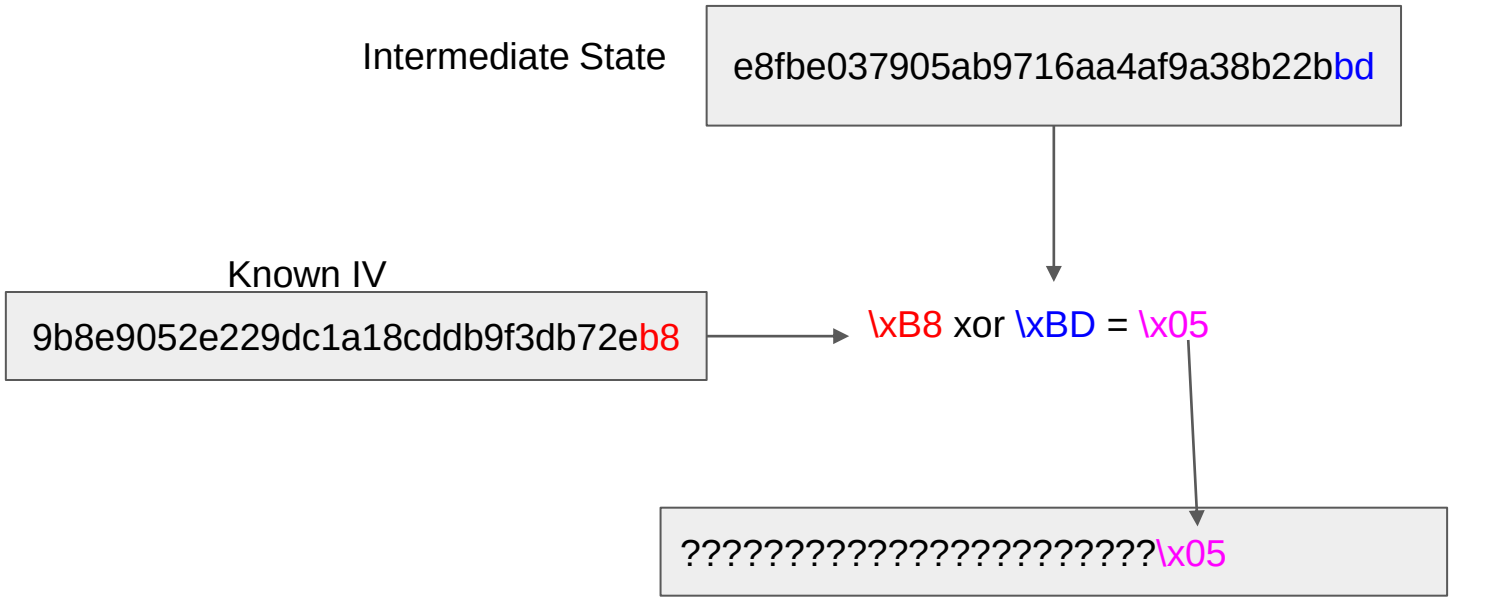


The actual attack, for real



Remember that AES is padded with easily determined numbers (0x01 means one byte, 0x02 means two bytes etc)

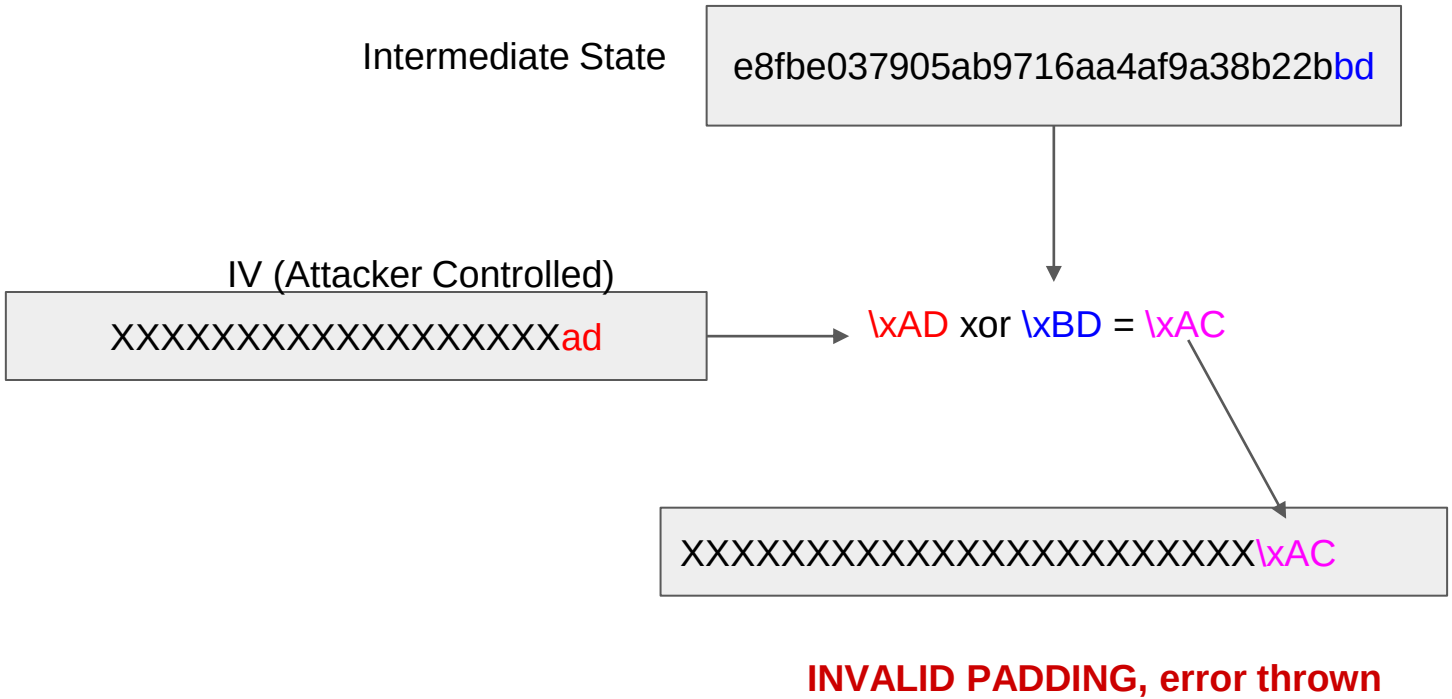
Nearly there!



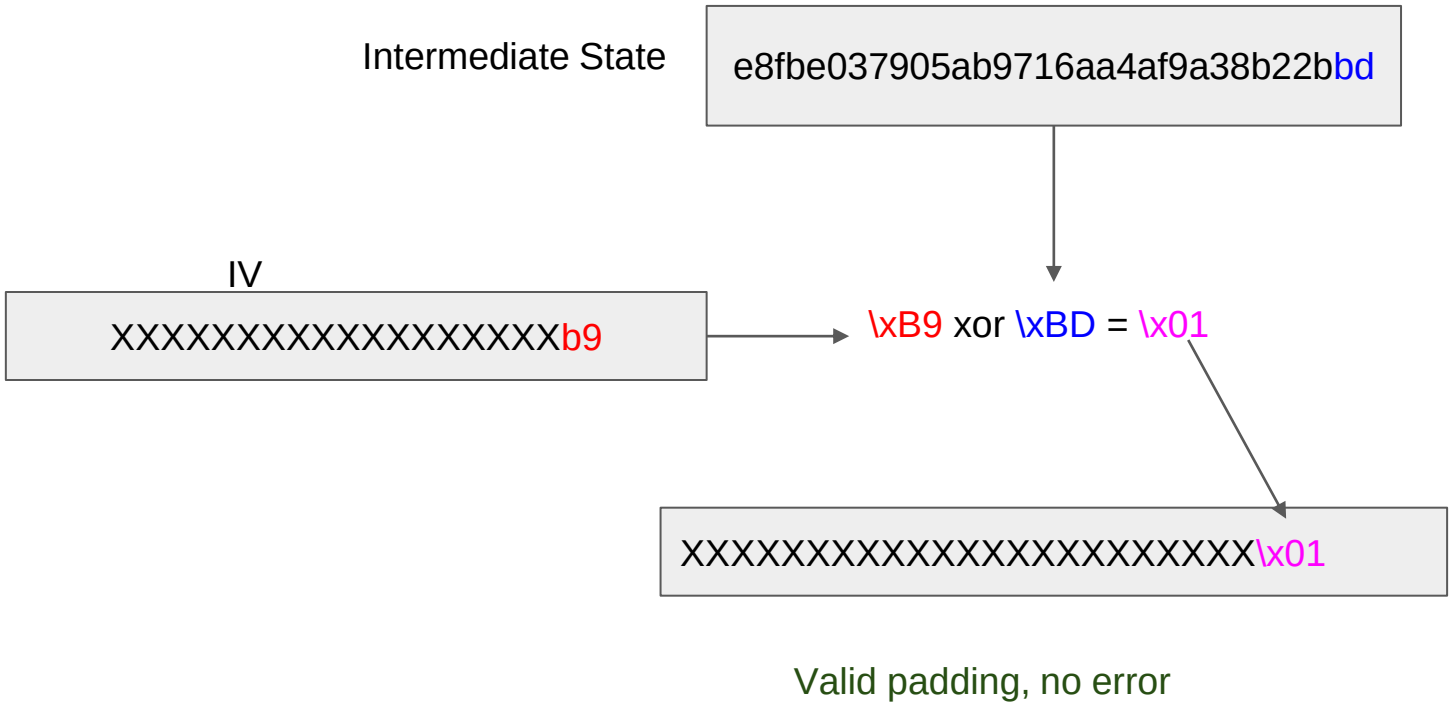
Valid padding, no error

Remember: We don’t see the result of decryption,
we just see if the padding worked or not

Nearly there!

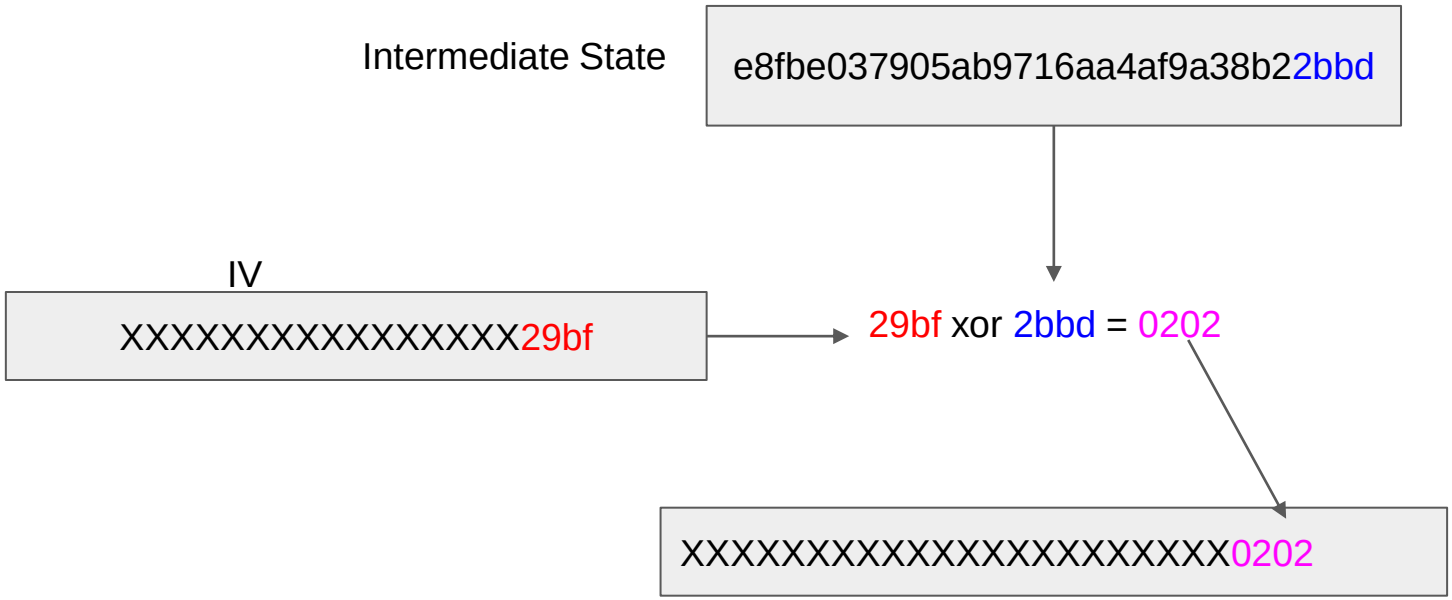


Nearly there!



XOR is commutative, so the attacker now knows: \xB9 xor \x01 = \xBD , which is a chunk of the intermediate state

Nearly there!



Valid padding, no error

The attacker repeats the last step, but with 0x02:

29bf xor 0202 = 2bbd , which is a chunk of the intermediate state

Finally...

- The attacker leaks the whole of the intermediate state using this technique, retrieving e8fbe037905ab9716aa4af9a38b22bbd
- The attacker then has two options:
  - if the IV used to encrypt was known (which it's usually not), then they simply XOR the IV with the intermediate state and the plaintext pops out.
  - If the IV is not known, the first block of ciphertext is used as the IV. Blocks 2-n can then be recovered, but the first block can't be
- In this case, the IV is known, so we xor the intermediate state with the known IV giving:
  e8fbe037905ab9716aa4af9a38b22bbd xor 9b8e9052e229dc1a18cddb9f3db72eb8 =
  7375706572736 56b7269740505050505
- We then unpad the result and take the ASCII decoding, producing the plaintext supersekrit

Attack done!

Questions?

Actually running the attack

Insert a live demo that fails here

Try it yourself!

- *Oracle, but not the TikTok kind* challenge on TCTF in the Modern Cryptography category
- 500 points

Proud Sponsors

CACI

EVER VIGILANT