Symmetric Encryption

Introduction to AES and ChaCha20

Jack P. (Master of Coin)

I am not an expert

I just like maths :)

Synopsis

(aka what are we talking about)

- What is a symmetric cipher?
- Understanding when symmetric ciphers are practical (and secure!)
- What are the differences between a block and stream cipher?
- When to use a stream cipher or a block cipher?
- Brief overview of popular symmetric ciphers: AES and ChaCha20
- Understanding limitations of “naked” symmetric ciphers
- Knowing which cipher-suite to use when securing data

What is a symmetric cipher?

- A symmetric cipher relies on a shared key to encrypt and decrypt data.

$$\text{Enc}(k, M) = C \text{ and } \text{Enc}^{-1}(k, C) = M$$
where k is a random bit vector and M is the plaintext

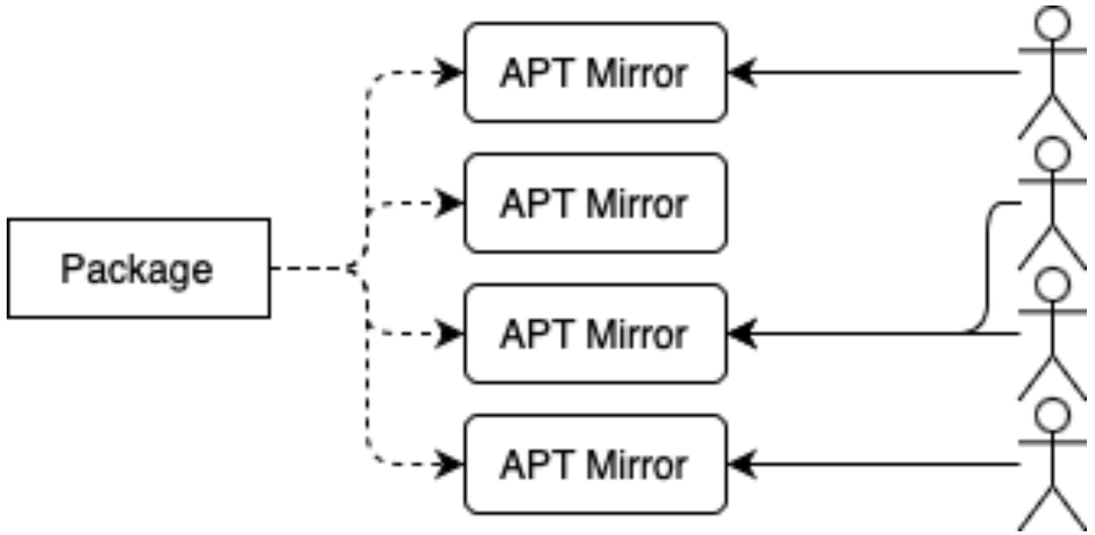
- The key is used for both encryption and decryption.

When are symmetric ciphers practical?

- If the data is static (e.g., encrypted hard-drive or encrypted backups).
- If the parties can securely transmit a key (e.g., PSK or via AKE).
- If the encryption must be performant (e.g., RTP).
- If the encryption must be efficient (e.g., encryption on embedded devices).

When are symmetric ciphers impractical?

- If data is communicated between unknown parties (e.g., package mirror and user).
- If only one peer should produce data (e.g., software updates).



Are symmetric ciphers secure?

- Symmetric ciphers are not invulnerable to attacks:
  TEA, XTEA, DES, Blowfish, RC4, etc.

- There are concerns regarding Post-Quantum-Cryptography (PQC):
  Grover’s algorithm

Not inherently. All cryptographic protocols must be designed with upgradability in mind to adapt as research advances.

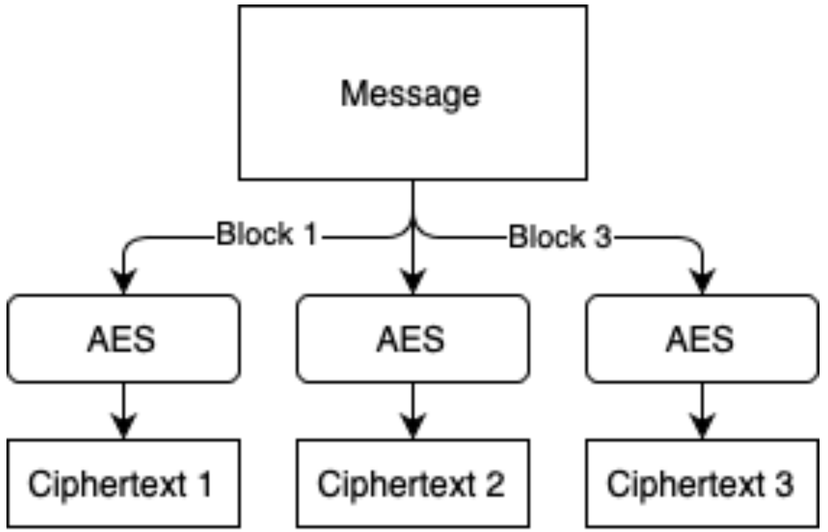
What is a block cipher and stream cipher?

- A block cipher encrypts data with a fixed key whereas a stream cipher encrypts using a continuous key stream.
- Block ciphers use a shared key to create an initial state which is mixed with the plaintext to produce ciphertext.
  (e.g., XOR-ing plaintext against a random key)
- Stream ciphers use a shared key to seed an initial state which produces a stream of bits which are mixed with the plaintext to produce the ciphertext.
  (i.e., seed a CSPRNG to generate a keystream)
- **The input length to a block cipher must be a multiple of the block size.**

What is AES?

Overview

- AES (“Rijndael”) is a block cipher developed for NIST in 1998.
- AES supports 128, 192, and 256-bit block sizes (applies to key sizes).
- AES has shown to be secure against all public research.



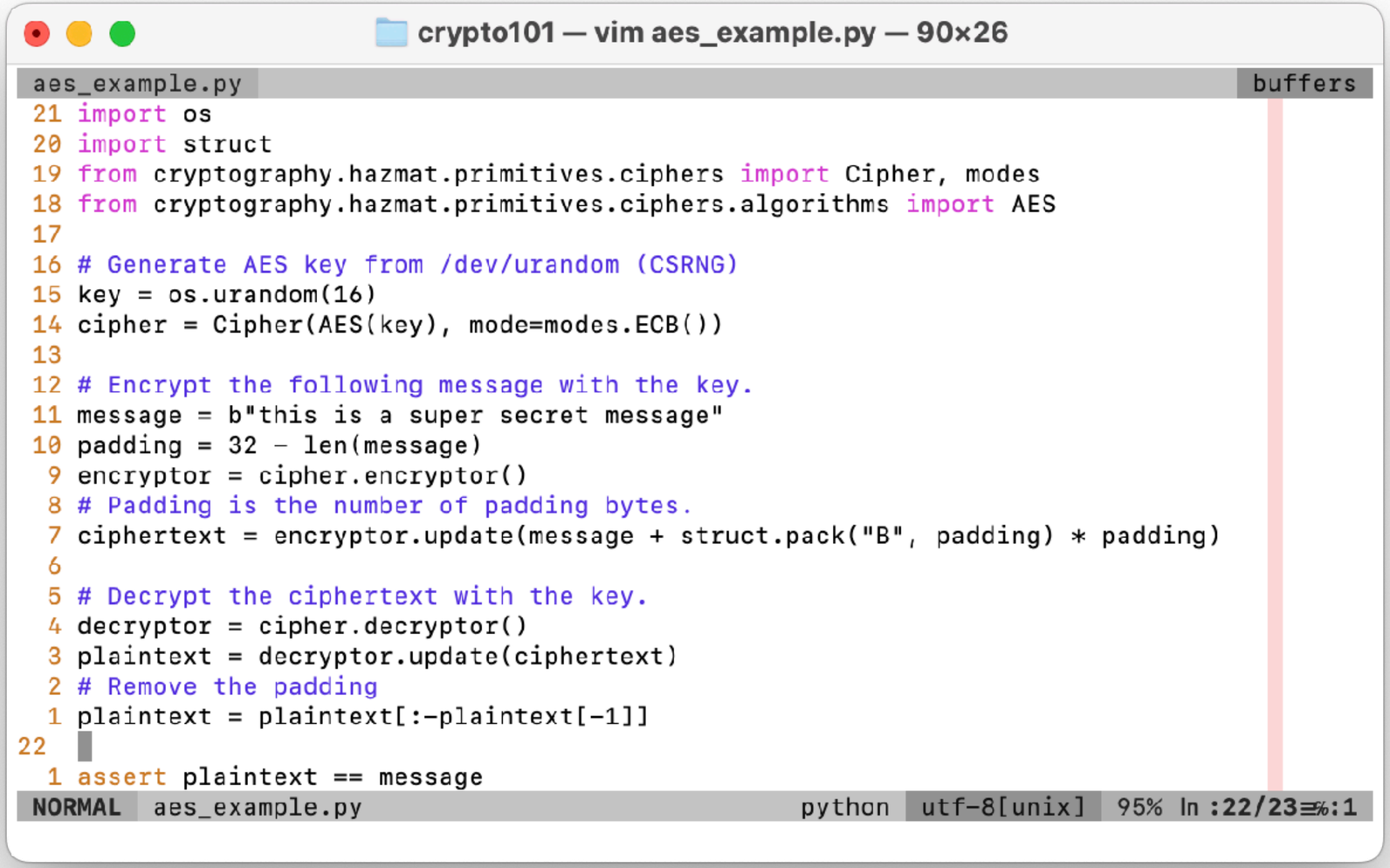
What is AES?

Design

- The input is split into block sized chunks which are independently encrypted. If the input length is incongruent with the block size then **the input is padded to the next full block.**
- AES performs a series of operations on 4x4 matrices (a block).
- AES implements a series of “rounds” which shuffle the matrix elements along the columns and rows, influenced by a round key derived from the input key.
- AES uses an S-box (substitution box) to swap elements of the matrix against a hard-coded table for *advanced maths reasons*.

What is AES?

Application



What is ChaCha20?

Overview

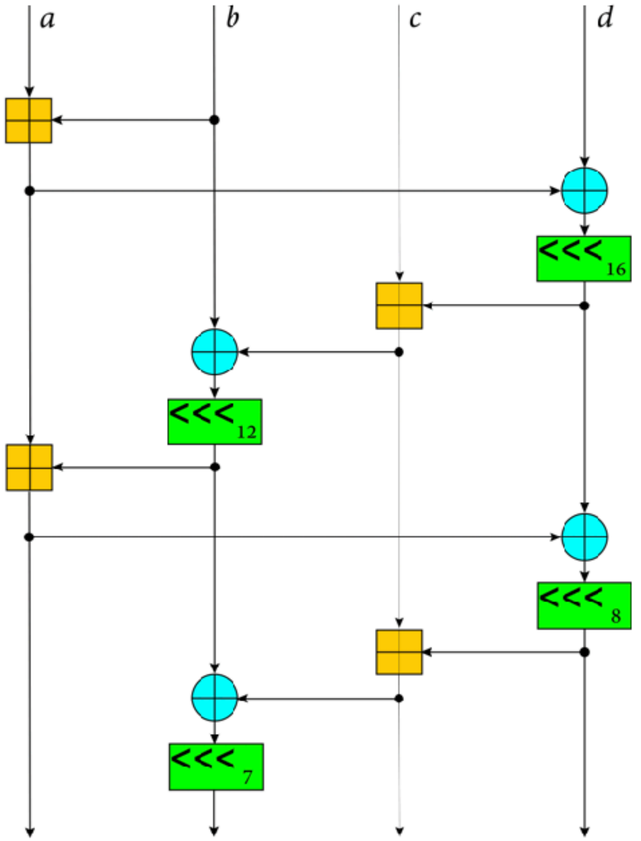
- ChaCha20 is a stream cipher designed by Daniel J. Bernstein in 2008 as an adaptation of Salsa20.
- ChaCha20 has 128 and 256 bit keys with a 512-bit state (for the stream).
- ChaCha20 has shown to be secure against all public research.

(the IETF has a proposal for XChaCha20 with better nonce misuse protection)

What is ChaCha20?

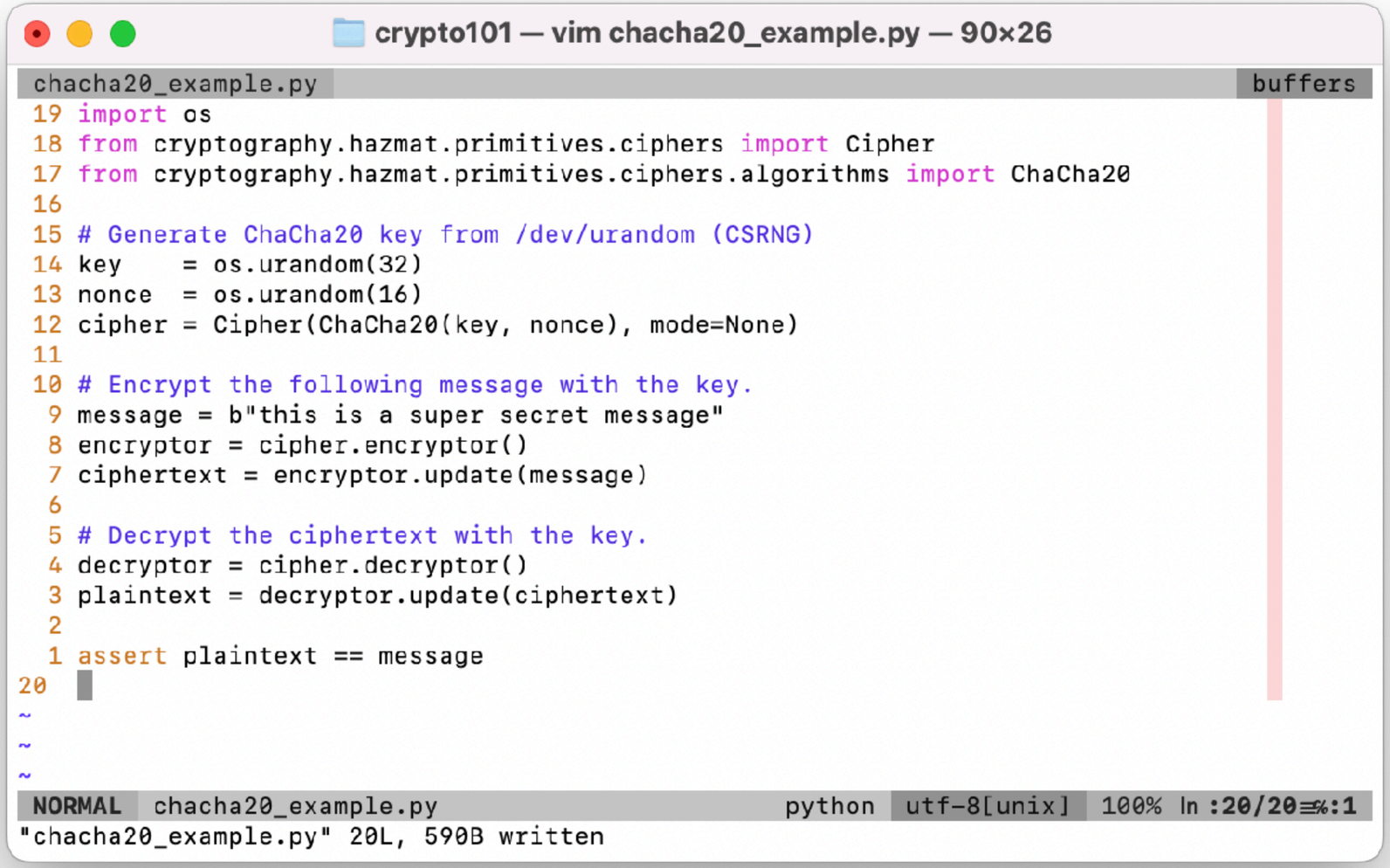
Design

- ChaCha20 generates a keystream from a nonce (we'll get to that), counter, and key.
- The keystream is updated for every block of input encrypted by ChaCha20.
- Each plaintext block is mixed against the current state to generate cipertext.



What is ChaCha20?

Application



ChaCha20 vs. AES

Which to choose and why

- Hardware accelerated AES is faster than software implementations of ChaCha20 (if you support AES-NI then AES is more efficient).
- Software implementations of AES are outperformed by ChaCha20.
- Legal compliance: FIPS 💩
- Using ChaCha20 to avoid AES S-box timing attacks.
- ChaCha20 has built-in nonce protection unlike raw AES.

Practical Stuff :)

AES distinguishability

Characterising the message plaintext from ciphertext.

- Clone the repository: https://github.com/gusecurity/crypto101
- Install dependencies:

```
$ python -m venv .venv
$ source .venv/bin/activate
$ pip install -r requirements.txt
```

- Read the script in `ex1/election.py` then determine who voted A or B. 60% voted for A and 40% voted for B.
- The “packet capture” of the voting machine is in `ex1/votes.txt`

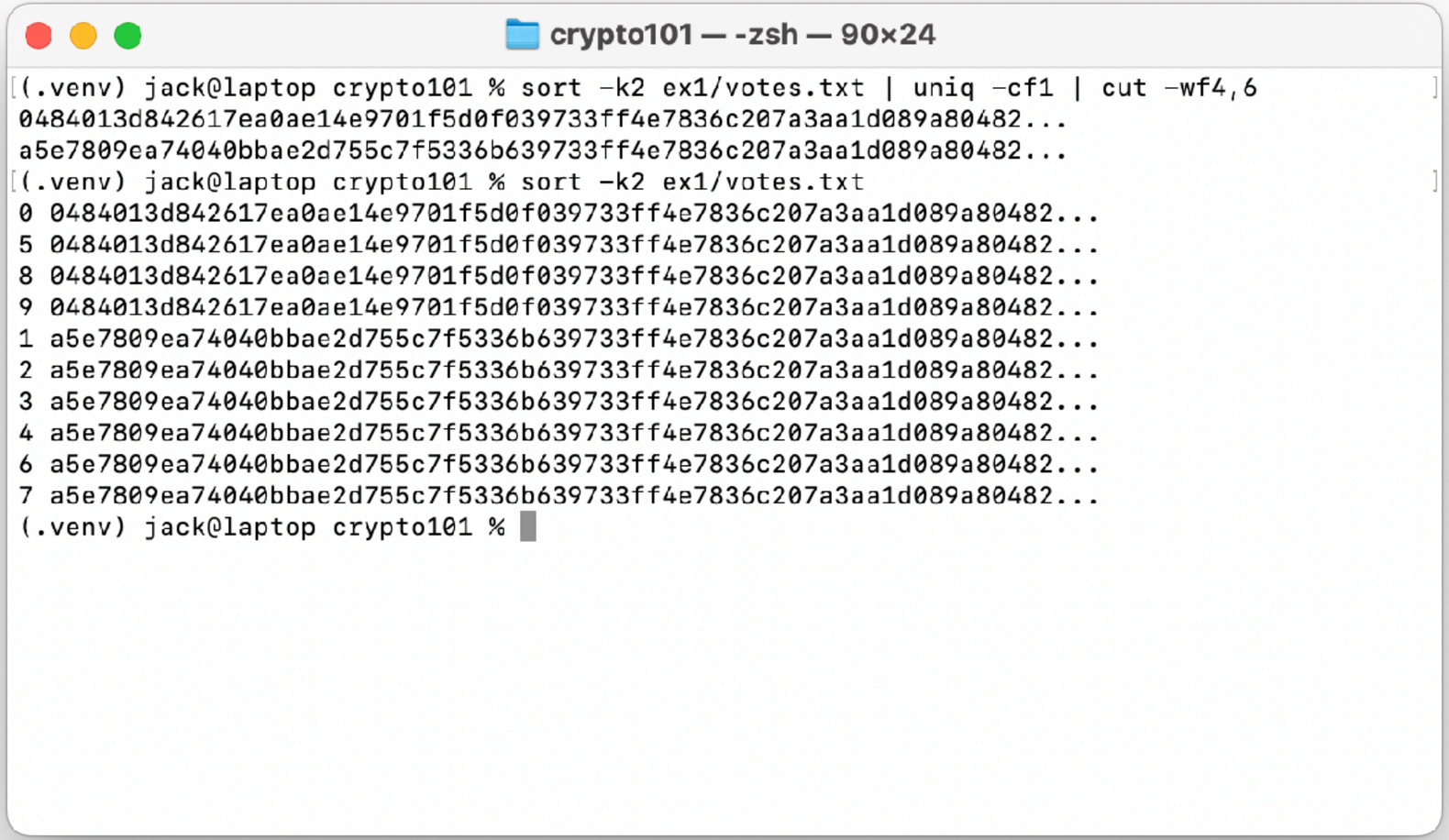
AES distinguishability

Characterising the message plaintext from ciphertext.

Voted for A	Voted for B
1	0
2	5
3	8
4	9
6	
7	

AES distinguishability

Characterising the message plaintext from ciphertext.



AES distinguishability

Characterising the message plaintext from ciphertext.

- This can be fixed by using a key per user (since we only encrypt a single block) but to decrypt a vote we must know who voted!
- **Using a cipher mode like CBC will not help since we have a single block!**
- Use paper ballots… but that’s too reasonable
- Use nonces!

AES distinguishability (nonces)

Characterising the message plaintext from ciphertext.

Not this one

AES distinguishability (nonces)

Characterising the message plaintext from ciphertext.

- Nonces are a way to scramble the ciphertext such that two identical blocks of plaintext correspond to distinct ciphertext blocks.
- **Read the standardisation material for your ciphersuite for proper usage.**
- Common implementations randomly generate the nonce or use a deterministic counter.
- For some algorithms, reusing a nonce can be catastrophic (see AESGCM and ECDSA).
- **We can use a vote counter as a nonce for the ciphertext.**

Ciphertext Malleability

Blind modifications to ciphertext to exploit applications.

- Use the same repository as before.
- Run the server with:
  `$ python ex2/server.py 8080`
- Read the script `ex2/server.py` then modify the captured ciphertext in `ex2/message.txt` to login as `WebAdmin` to get the flag.
- If you are not confident with Python then feel free to adapt `ex2/partial_sol.py`.

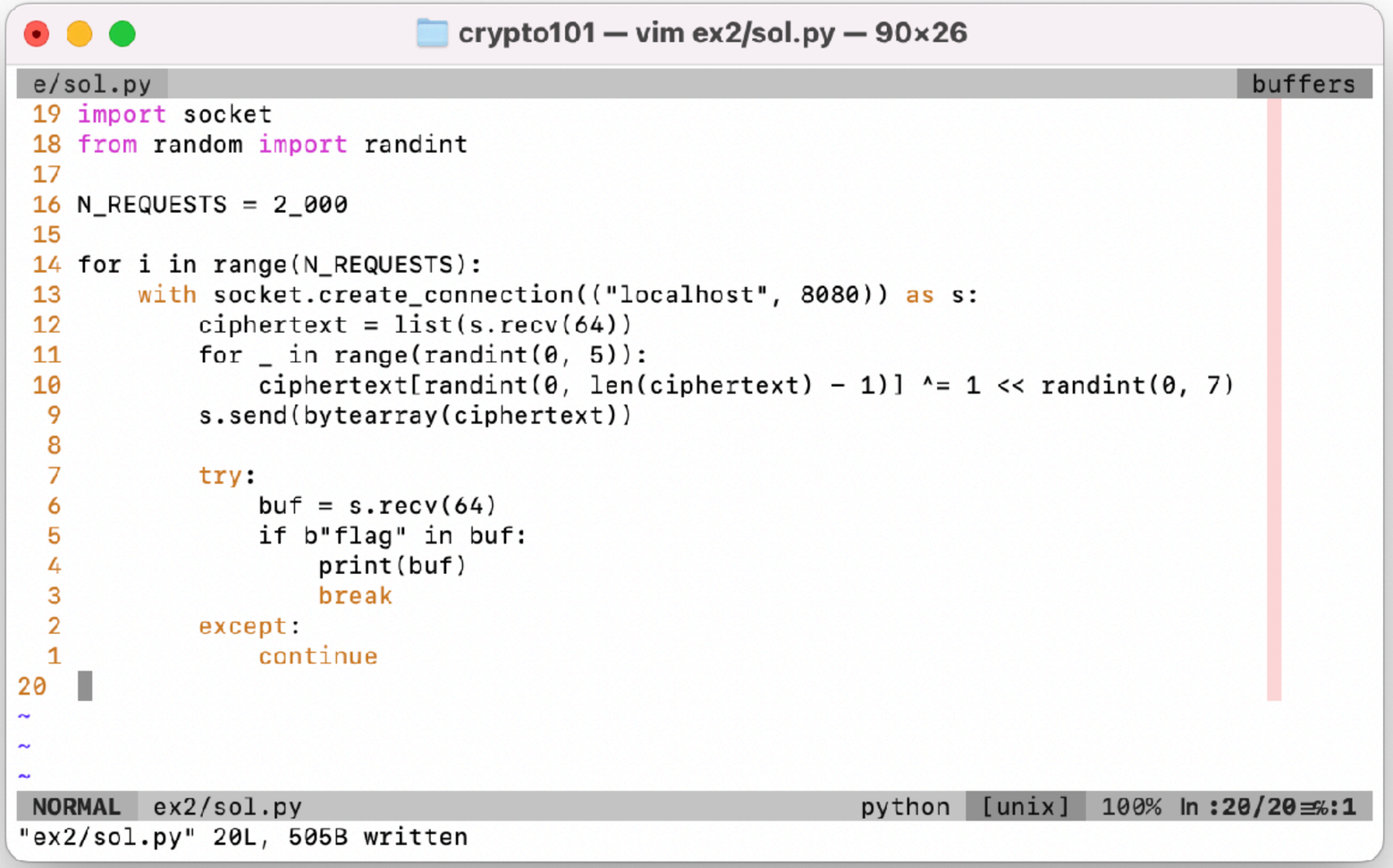
Ciphertext Malleability

Blind modifications to ciphertext to exploit applications.

The flag is in the source 5head.

Ciphertext Malleability

Blind modifications to ciphertext to exploit applications.



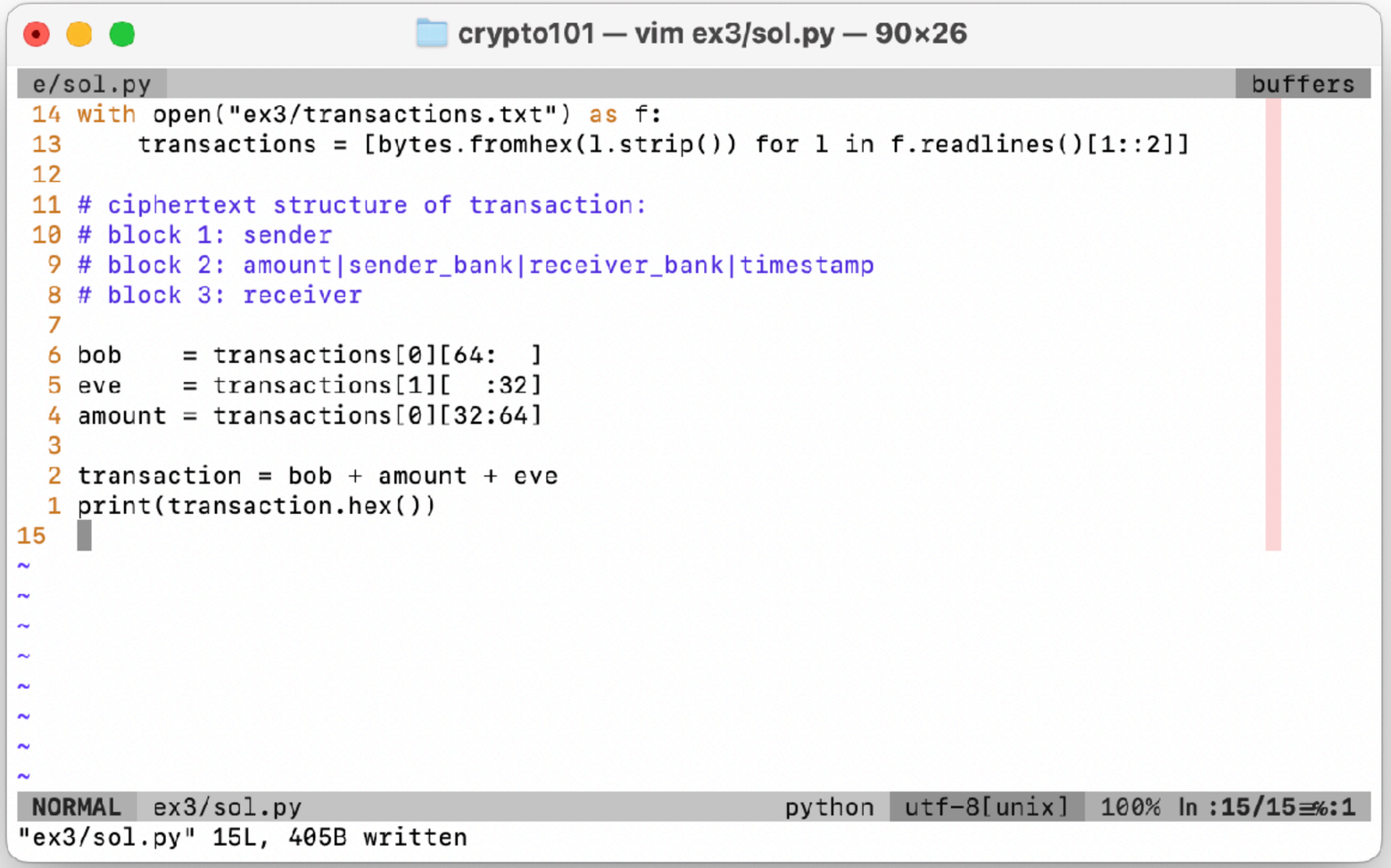
Ciphertext Juggling

I have no idea what the proper term is.

- The third exercise is the most subtle (and appears in real scenarios, such as NFC payments).
- The bank teller sends transactions to a central banking service.
- Figure out a way to transfer £500 from Alice to Eve from the captured transactions in `ex3/transactions.txt`.
- The script `ex3/bank.py` is responsible for creating the transactions.
- **This is hard so please ask for hints if you are stuck.**
- **The captured messages use a different key so the correct solution will not work!**

Ciphertext Juggling

I have no idea what the proper term is.



Optional Exercise (Advanced)

Cipher Modes

WTF is this CBC thing?

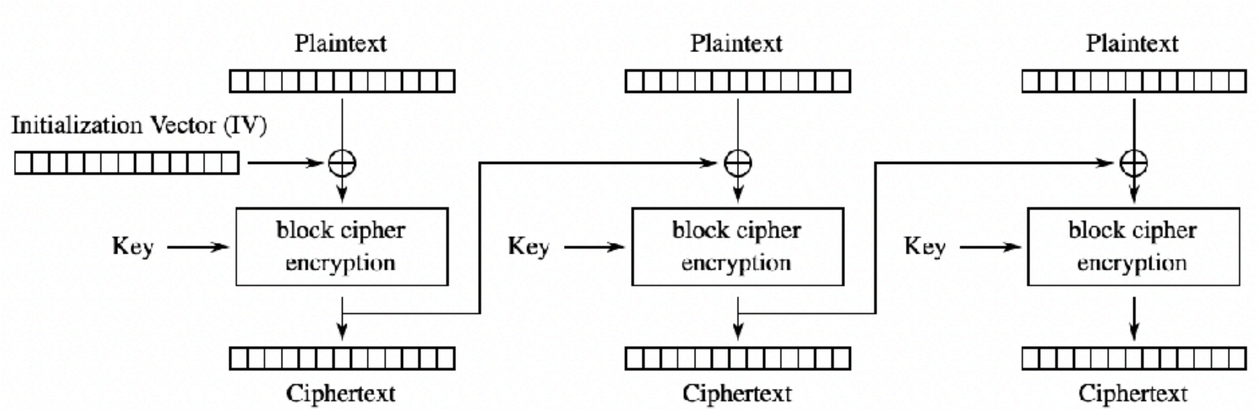
- Block ciphers have modes designed to scramble the ciphertext to prevent reliable malleability and distinguishing ciphertext.
- Cipher modes can be thought of as a way to transform a primitive cipher like AES by adding desirable properties (such as the above).
- Not all cipher modes are perfect: CBC does not have parallelised encryption.
- **Additional functionality can introduce insecurities.**

AES-CTR can turn AES into a pseudo-stream cipher!

Cipher Modes

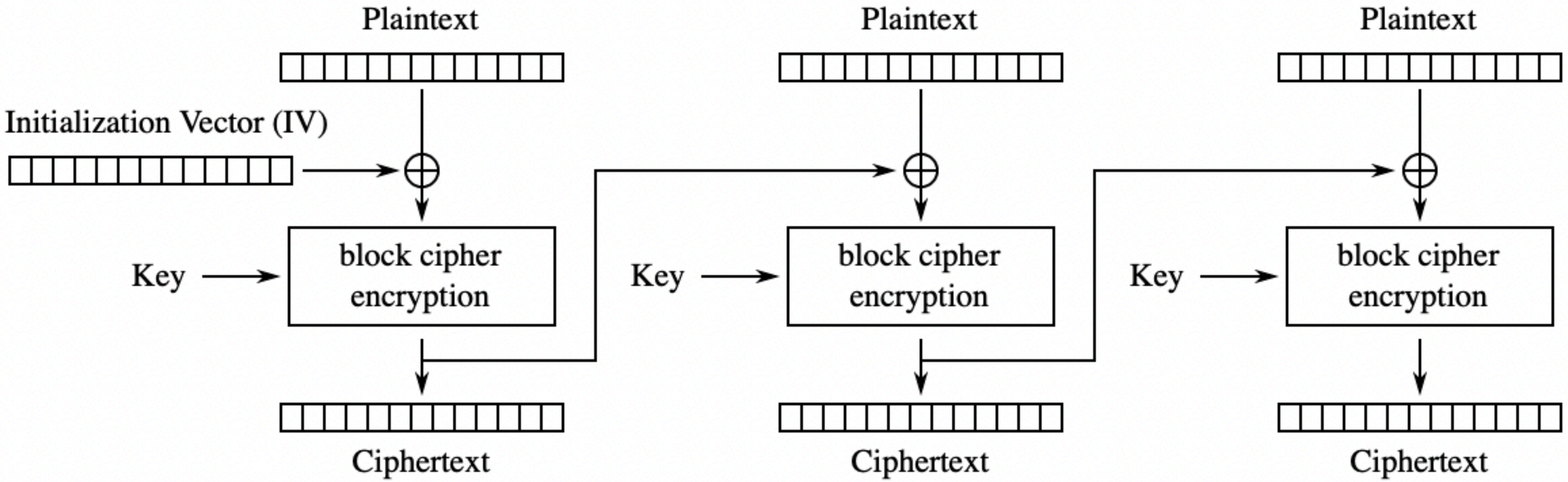
WTF is this CBC thing?

- CBC (Cipher-Block-Chaining) uses the previous ciphertext block to scramble the subsequent block by XOR-ing the plaintext.
- CBC implements padding by padding up to the next full block.
  If the plaintext is a multiple of the block size then a block of padding is added.
- The padding is the number of padding bytes appended to the **plaintext (PKCS#7).**



Cipher Modes

WTF is this CBC thing?



Padding Oracle

You thought CBC could save you?

- The final exercise is a walkthrough showing how to leak the plaintext of the message between Alice and Bob.
- The `ex4/decryptor.py` script will attempt to decrypt an input message.
- The captured message between Alice and Bob is in `ex4/message.txt`.
- Partial solution in `ex4/partial_sol.py`.
- **Feel free to give it a shot if you feel confident.**

Padding Oracle

You thought CBC could save you?

- This attack is online and requires interaction with the decryption oracle.
- This can be prevented using **authentication.**

- **Note: padding oracle can serve as an *encryption oracle* (CBC-R).**

Summary

Authentication

(and nonces)

Epilogue
(the boring bit)

- Please participate in the poll on the Discord channel to vote for the next workshop topic!
- If you have any suggestions then reach out to any of the committee members.
- If I could have done better then let me know so I can improve :)

If you enjoyed this topic then I am happy to do future presentations on other cryptographic concepts (and even cryptanalysis).