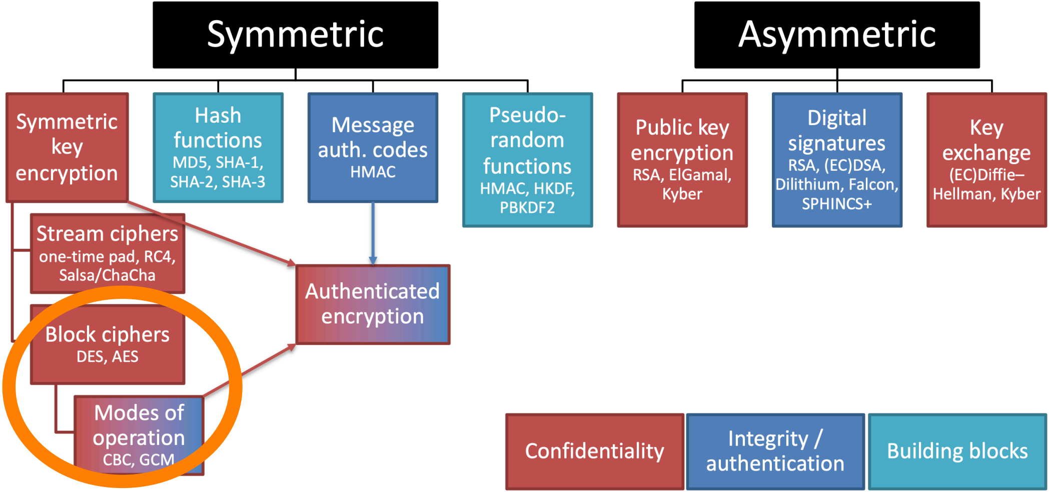
Topic 1.4
Symmetric encryption – Block cipher modes of operation

Douglas Stebila

CO 487/687: Applied Cryptography

Fall 2024

Map of cryptographic primitives



Symmetric

Symmetric key encryption

Stream ciphers
one-time pad, RC4, Salsa/ChaCha

Block ciphers
DES, AES

Modes of operation
CBC, GCM

Hash functions
MD5, SHA-1, SHA-2, SHA-3

Message auth. codes
HMAC

Authenticated encryption

Pseudo-random functions
HMAC, HKDF, PBKDF2

Asymmetric

Public key encryption
RSA, ElGamal, Kyber

Digital signatures
RSA, (EC)DSA, Dilithium, Falcon, SPHINCS+

Key exchange
(EC)Diffie–Hellman, Kyber

Confidentiality

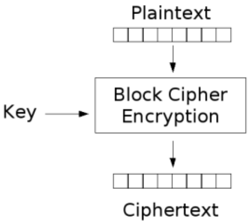
Integrity / authentication

Building blocks

Block ciphers vs. stream ciphers

Recall:

- A stream cipher is a symmetric-key encryption scheme in which each a pseudorandom sequence of arbitrary length is generated to encrypt successive character of plaintext of ciphertext.
- A block cipher is a symmetric-key encryption scheme in which a fixed-length block of plaintext determines an equal-sized block of ciphertext.



Encrypting bulk data

What if one needs to encrypt large quantities of data?

- With a stream cipher, just encrypt each character.
- With a block cipher, there are some complications if:
  - the input is larger than one block, or
  - the input does not fill an integer number of blocks.

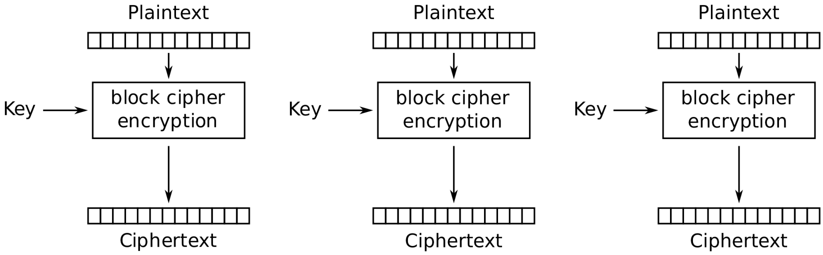
To deal with these problems, we use a *mode of operation*, which means a specification for how to encrypt multiple and/or partial data blocks using a block cipher.

Padding

- Some modes, namely ECB and CBC, require the plaintext to consist of one or more complete blocks.
- NIST Special Publication 800-38A suggests a padding method as follows.
    1. append a single ‘1’ bit to the data string
    2. pad the resulting string by as few ‘0’ bits, possibly none, as are necessary to complete the final block.
- The padding bits can be removed unambiguously, if the receiver knows that this padding method is used.
    1. remove all trailing ‘0’ bits after the last ‘1’ bit
    2. remove a single ‘1’ bit.

Electronic Codebook (ECB) mode

The obvious approach is to encrypt each ℓ bits independently, where ℓ is the block size.



Electronic Codebook (ECB) mode encryption

All figures from `https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation`

Electronic Codebook (ECB) mode

ECB is the most basic mode of a block cipher.

Encryption

- $C_i = E(K, P_t)$
- Plaintext block P_i is encrypted with the key K to produce ciphertext block C_i

Decryption

- $P_i = D(K, C_i)$
- Ciphertext block C_i is decrypted with the key K to produce plaintext block P_i

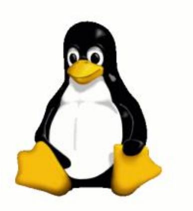
Problems with ECB mode

Although stream ciphers are (usually) secure when used in the obvious way, block ciphers in ECB mode are INSECURE!

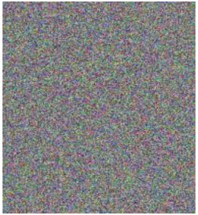
- A block cipher, unlike a stream cipher, is stateless.
- ECB mode is equivalent to a giant substitution cipher where each ℓ-bit block is a “character”
- Semantic security is immediately violated: One can tell by inspection whether or not two blocks of ciphertext correspond to identical plaintext blocks (violates “no partial information”)

ECB example

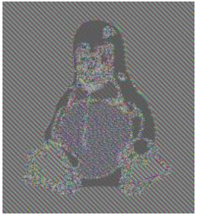
Original



Preferred



ECB mode



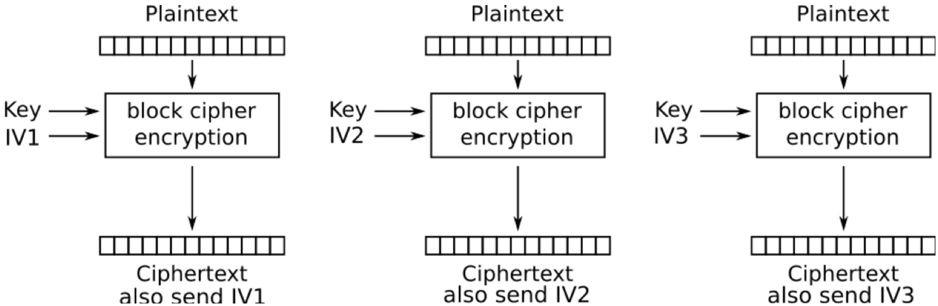
ECB mode properties

Randomised	✘
Padding	Required
Error propagation	Errors propagate within blocks
IV	None
Parallel encryption?	✓
Parallel decryption?	✓
IND-CPA secure?	✘

Because it is deterministic, ECB mode is not normally used for bulk encryption

Adding IVs

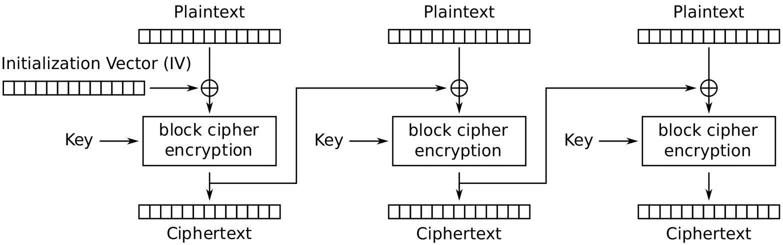
Idea: Use a new (non-secret) initialization vector for each block of plaintext.



Problem: have to send an IV for each block of ciphertext, adding *linear* overhead.

Cipher Block Chaining (CBC) mode

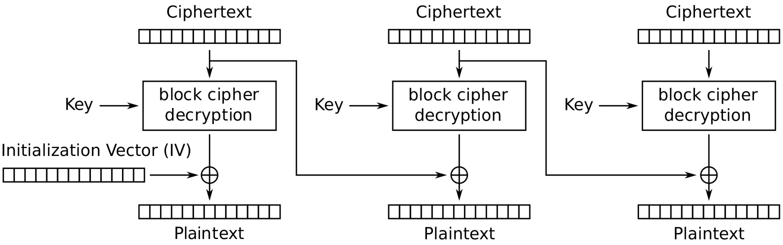
Idea: Use a single (non-secret) initialization vector for the first block of plaintext, and "chain" the (pseudorandom) ciphertext blocks as the next blocks' initialization vectors. The IV is included as part of the ciphertext.



Cipher Block Chaining (CBC) mode encryption

Cipher Block Chaining (CBC) mode

Idea: Use a single (non-secret) initialization vector for the first block of plaintext, and "chain" the (pseudorandom) ciphertext blocks as the next blocks' initialization vectors. The IV is included as part of the ciphertext.



Cipher Block Chaining (CBC) mode decryption

CBC mode

- CBC “chains” the blocks together.
- A random initialisation vector IV is chosen and sent together with the ciphertext blocks.

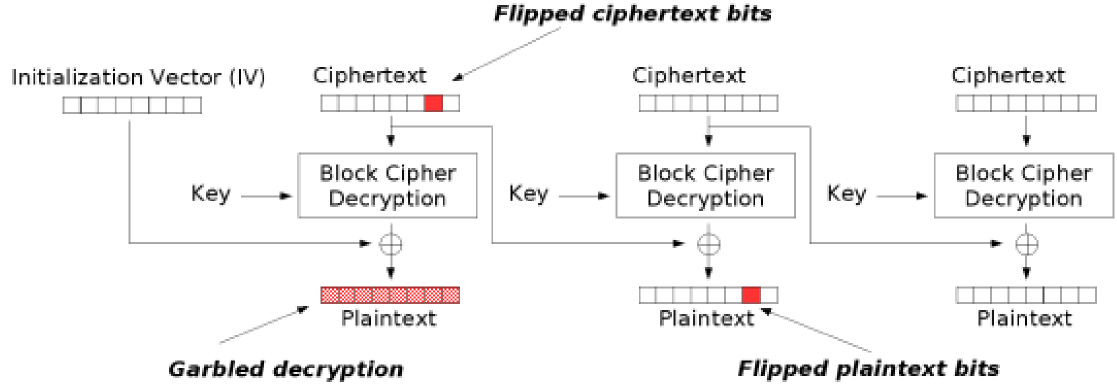
Encryption

- $C_i = E(K, P_i \oplus C_{i-1})$, where C_0 = IV.
- P_i is XOR’d with the previous ciphertext block C_{i-1}, and encrypted with key K to produce ciphertext block C_i. For the first plaintext block IV is used for the value C_0.

Decryption

- $P_i = D(K, C_i) \oplus C_{i-1}$, where C_0 = IV.
- C_i is decrypted with the key K, and XOR’d with the previous ciphertext block C_{i-1} to produce plaintext block P_i. As in encryption, IV is used in place of C_0.

CBC mode error propagation



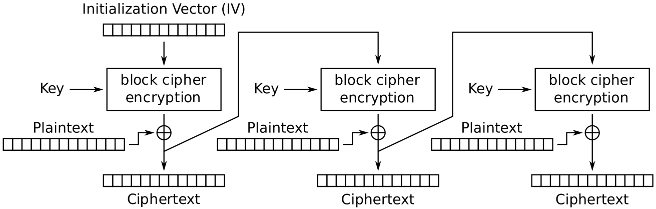
Modification attack or transmission error for CBC

CBC mode properties

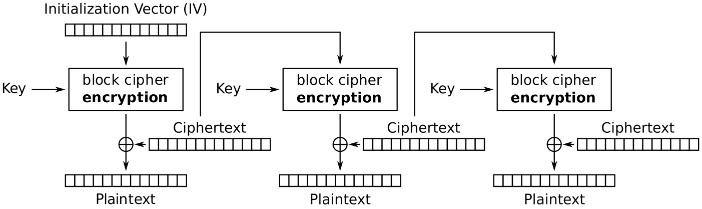
Randomised	✓
Padding	Required
Error propagation	Errors propagate within blocks and into specific bits of next block
Dropped blocks	Can’t decrypt next block, can decrypt subsequent blocks
IV	Must be random
Parallel encryption?	✘
Parallel decryption?	✓
IND-CPA secure?	✓

CBC mode is commonly used for bulk encryption and is supported in most libraries and protocols.

Cipher Feedback (CFB) mode



Cipher Feedback (CFB) mode encryption


Initialization Vector (IV)
Key
block cipher **encryption**
Ciphertext
Plaintext
Key
block cipher **encryption**
Ciphertext
Plaintext
Key
block cipher **encryption**
Ciphertext
Plaintext

Cipher Feedback (CFB) mode decryption

Cipher Feedback (CFB) mode

CFB “feeds” the ciphertext block back into the enciphering/deciphering process, thus “chaining” the blocks together.

Encryption

- $C_i = E(K, C_{i-1}) \oplus P_i$, where $C_0 = IV$

Decryption

- $P_i = E(K, C_{i-1}) \oplus C_i$, where $C_0 = IV$

Propagation of channel errors:

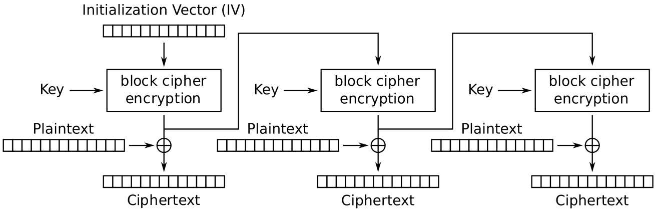
- a one-bit change in C_i produces a one-bit change in P_i, and complete corruption of P_{i+1}

CFB mode properties

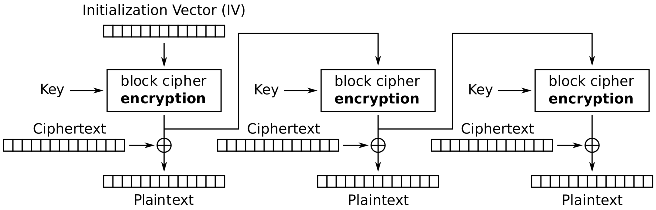
The underlying block cipher is only used in **encryption** mode.

Randomised	✓
Padding	Not required
Error propagation	Errors occur in specific bits of current block and propagate into next block
IV	Must be random
Parallel encryption?	✘
Parallel decryption?	✓
IND-CPA secure?	✓

Output Feedback (OFB) mode



Output Feedback (OFB) mode encryption


Initialization Vector (IV)
Key
block cipher **encryption**
Ciphertext
Plaintext
Key
block cipher **encryption**
Ciphertext
Plaintext
Key
block cipher **encryption**
Ciphertext
Plaintext

Output Feedback (OFB) mode decryption

Output Feedback (OFB) mode

- OFB “feeds” the output block back into enciphering/deciphering process.
- OFB is, in effect, a *synchronous stream cipher*. The keystream is:

$$O_i = E(K, O_{i-1}),$$

where $O_0 = IV$ is chosen at random.

Encryption

- $C_i = O_i \oplus P_i$

Decryption

- $P_i = O_i \oplus C_i$

- *Propagation of channel errors:*
  - a one-bit change in the ciphertext produces a one-bit change in the plaintext at the same location

OFB mode properties

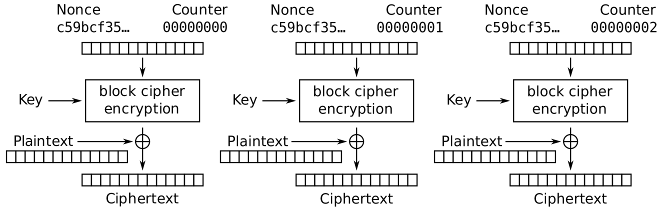
The underlying block cipher is only used in **encryption** mode.

Randomised	✓
Padding	Not required
Error propagation	Errors occur in specific bits of current block
IV	Must be unique
Parallel encryption?	✘ (but keystream can be computed in advance)
Parallel decryption?	✘ (but keystream can be computed in advance)
IND-CPA secure?	✓

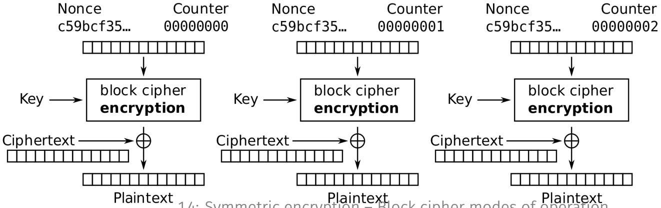
OFB mode is a synchronous stream cipher mode.

Counter (CTR) mode

Choose a nonce at random during encryption. Prepend the nonce to the ciphertext.



Counter (CTR) mode encryption



Counter (CTR) mode

- CTR is a *synchronous stream cipher*. The keystream is generated by encrypting successive values of a "counter", initialised using a nonce (randomly chosen value) N:

$$O_i = E(K, T_i),$$

where $T_i = N \| i$ is the concatenation of the nonce and block number i.

Encryption

- $C_i = O_i \oplus P_i$

Decryption

- $P_i = O_i \oplus C_i$

- *Propagation of channel errors:*
  - a one-bit change in the ciphertext produces a one-bit change in the plaintext at the same location

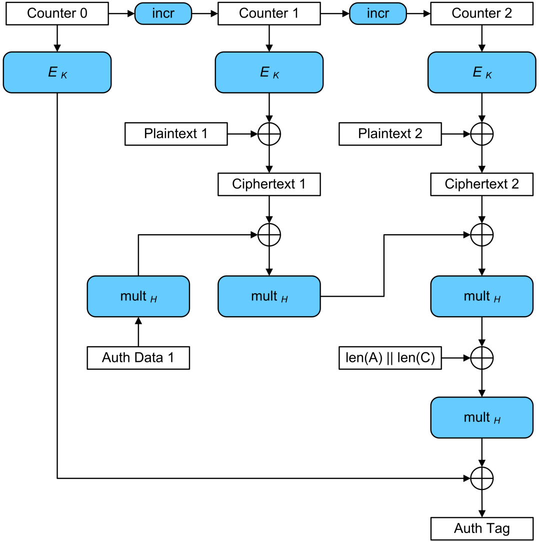
CTR mode properties

The underlying block cipher is only used in **encryption** mode.

Randomised	✓
Padding	Not required
Error propagation	Errors occur in specific bits of current block
IV	Nonce must be unique
Parallel encryption?	✓
Parallel decryption?	✓
IND-CPA secure?	✓

- CTR mode is a synchronous stream cipher mode.
- CTR mode is also good for access to specific plaintext blocks without decrypting the whole stream.

Preview: Authenticated encryption (Galois Counter Mode)

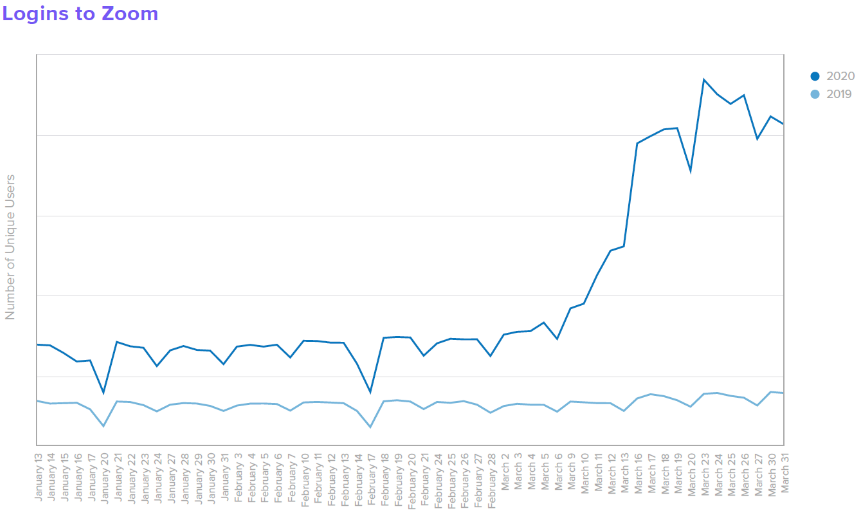


Summary of mode properties

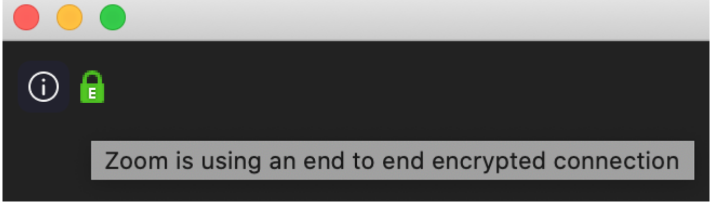
	ECB	CBC	CFB	OFB	CTR
IND-CPA secure?	✗	✓	✓	✓	✓
Randomised	✗	✓	✓	✓	✓
Padding required?	✓	✓	✗	✗	✗
Error propagation within block? next? subsequent?	✓✗✗	✓✓✗	✓✓✗	✓✗✗	✓✗✗
Dropped block decrypt next? subsequent?	✓✓	✗✓	✗✓	✓* ✓*	✓* ✓*
IV	none	random	unique	unique	unique
Parallel encryption?	✓	✗	✗	✗	✓
Parallel decryption?	✓	✓	✓	✗	✓
Pre-compute encryption?	✗	✗	✗	✓	✓
Pre-compute decryption?	✗	✗	✗	✓	✓

* if receiver realizes block has been dropped and advances appropriately.

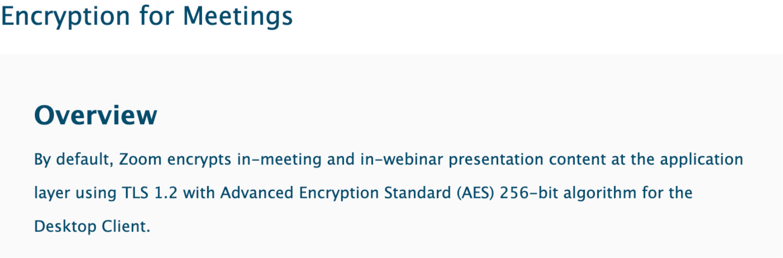
Zoom



`https://blog.zoom.us/okta-businesses-work-from-home-report-zoom-the-fastest-growing-app-march-2020/zoom-logins/`



Zoom is using an end to end encrypted connection



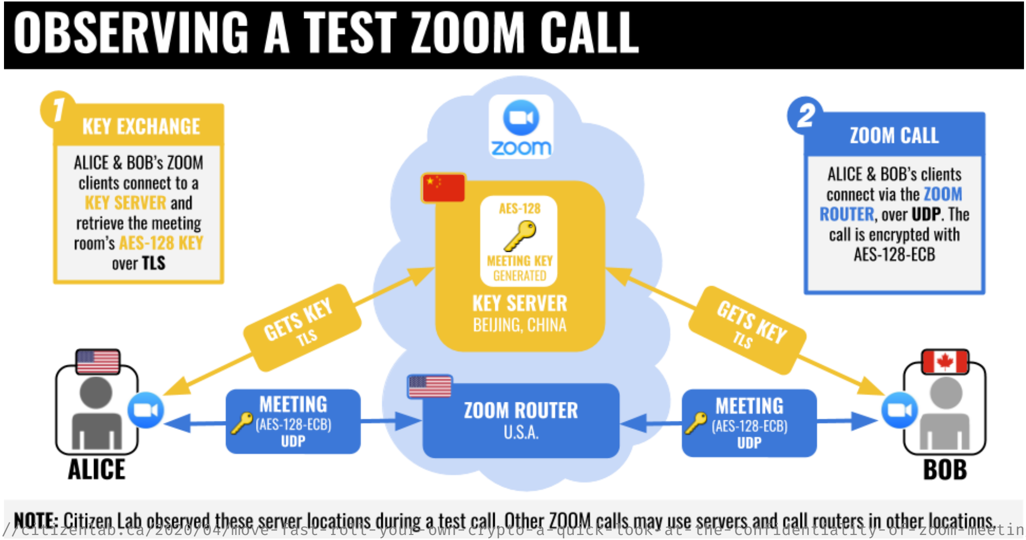
Encryption for Meetings

Overview

By default, Zoom encrypts in-meeting and in-webinar presentation content at the application layer using TLS 1.2 with Advanced Encryption Standard (AES) 256-bit algorithm for the Desktop Client.

`https://citizenlab.ca/2020/04/move-fast-roll-your-own-crypto-a-quick-look-at-the-confidentiality-of-zoom-meetings/`

Zoom as of April 2020



OBSERVING A TEST ZOOM CALL

1 KEY EXCHANGE

ALICE & BOB's ZOOM clients connect to a KEY SERVER and retrieve the meeting room's AES-128 KEY over TLS

2 ZOOM CALL

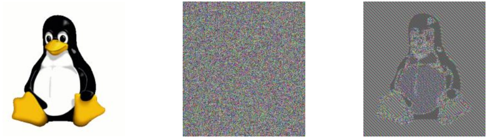
ALICE & BOB's clients connect via the ZOOM ROUTER, over UDP. The call is encrypted with AES-128-ECB

NOTE: Citizen Lab observed these server locations during a test call. Other ZOOM calls may use servers and call routers in other locations.

https://citizenlab.ca/2020/04/move-fast-roll-your-own-crypto-a-quick-look-at-the-confidentiality-of-zoom-meetings/

Zoom's problems as of April 2020

1. Not end-to-end encrypted. Key server knows the encryption key used for encryption of every meeting. Even if they claim not to use it to decrypt, they have the capability to do so if desired or ordered by a government authority to do so.
   - Contrast with WhatsApp/Signal, iMessage which both use Diffie–Hellman key exchange to avoid having servers know the encryption key.
2. Not actually using AES-256 as claimed. Observed to be using AES-128 for encrypting meeting audio/video.
3. Using AES-128 in ECB mode.



Is Zoom Fixed?

- April 1, 2020: [1] Zoom posts blog post with some details about encryption
- April 3, 2020: [2] Citizen Lab report detailing above problems released
- April 22, 2020: [3,4] Zoom announces version 5.0 which includes:
  - AES-256-GCM for audio/video encryption.
  - User control over which regions a call is routed through.
  - Other non-cryptographic security features (mostly to reduce Zoom-bombing: meeting passwords required by default, waiting room on by default)
  - Still not end-to-end encryption.

[1] https://blog.zoom.us/facts-around-zoom-encryption-for-meetings-webinars/
[2] https://citizenlab.ca/2020/04/move-fast-roll-your-own-crypto-a-quick-look-at-the-confidentiality-of-zoom-meetings/
[3] https://blog.zoom.us/zoom-hits-milestone-on-90-day-security-plan-releases-zoom-5-0/
[4] https://zoom.us/docs/doc/ZoomEncryptionWhitepaper.pdf

Is Zoom Fixed?

- June 2020: [5] Zoom announces plan for end-to-end encryption
- October 2020: [6] Zoom rolls out end-to-end encryption
    - Uses public key cryptography (Diffie–Hellman key exchange) to establish encryption keys between users without Zoom knowing them
    - Not all features available to calls using E2E encryption
    - No authentication of users
- August 2022: [7] Customer managed keys for enterprise users

[5] https://blog.zoom.us/end-to-end-encryption-update/
[6] https://blog.zoom.us/zoom-rolling-out-end-to-end-encryption-offering/
[7] https://blog.zoom.us/zoom-customer-managed-key/