Detecting Encryption Vulnerabilities with Lossless Compression

Richard Hansen
rhhansen@captechu.edu
Capitol Technology University
11301 Springfield Road, Laurel, MD 20708

ABSTRACT: Ciphertext entropy is a key property for detecting the use of insecure encryption modes such as Electronic Code Book (ECB) and for testing symmetric=key cryptographic algorithms. This paper discusses the use of lossless compression utilities as a proxy measurement for entropy and its use in detecting a limited set of encryption vulnerabilities. We also note the use of off-the-shelf algorithms yields a teaching tool for Information Technology and Cybersecurity students. Finally, small and consistent differences between encryption modes provide the potential to identify the encryption algorithm from compression factors.

Keywords: Cryptography, Cryptology, Encryption, Entropy, Compression, Cybersecurity, Vulnerability Assessment, Information Assurance, ECB

INTRODUCTION

In April, 2020 it was revealed that Zoom Meetings provided misleading information to clients regarding the encryption algorithm used for protection of confidentiality (Marczak and Railton, 2021). An encryption mode with known vulnerabilities, Electronic Code Book or ECB, was used without being disclosed to users and an unknown number of meetings were compromised/

This mode may be in use by other systems and applications. Developers, IT staff and the user community may not understand issues related to algorithms, modes and other encryption features (Bai et al., 2020). One potential method for detecting the use of ECB is measuring a property known as entropy for the output of the encryption process, the ciphertext. Entropy measurements can be useful for detecting a limited number of vulnerabilities including the use of ECB. Entropy measures the lack of order and predictability in a file; a low entropy measurement means there are patterns that can be detected or predicated and a high entropy measurement means there are few patterns that can be detected or predicted.

Compression is also used to identify patterns in data such as repeated characters or sequences of characters that have a mathematical relationship. The identified patterns are used to reduce the size of data so it will use less resources when stored or transmitted.

This paper details a qualitative study that provides the following contributions:

- Experimental results that validate the use of lossless data compression to detect the use of ECB mode in several algorithms, and detection of a limited number of issues related to poorly chosen encryption keys and initialization vectors.

- Experimental results that establish a relationship between measured entropy of ciphertext generated using symmetric key algorithms and the degree of lossless data compression possible with the commonly available GNU Zip utility.
- Techniques for use of lossless data compression as a teaching tool for the concepts of entropy as it relates to encryption/

The Problem – Detection of Vulnerable ECB Mode Encryption

Zoom Meetings claimed to use strong encryption to protect the confidentiality of meetings based on the Advanced Encryption Standard algorithm with a 256-bit key length (AES-256). In reality Zoom used a weaker AES variant, AES-128 in the Electronic Code Book (ECB) mode. ECB mode is vulnerable to exploitation when used for large amounts of data. Its use may be detected using specially selected inputs, known as plaintext, and entropy measurements of the resulting ciphertext output. Other problems may cause issues with encryption algorithms and their software implementations, such as values for keys and initialization vectors that are easily guessed or that produce undesired results.

Entropy refers to the lack of predictability in ciphertext. Ideally this means that knowledge of the previous bits and bytes in a file or stream of data does not allow prediction of the value of the next bits or bytes. Useful patterns should be obfuscated by a properly designed and implemented encryption algorithm which produces ciphertext with high degree of entropy (unpredictability).

Modern encryption algorithms such as the Advanced Encryption Standard (AES) use the principles of confusion and diffusion to obfuscate patterns in data and increase entropy. These terms originated with Claude Shannon's then-classified paper "A Mathematical Theory of Cryptography" (Shannon, 1945). Confusion refers to representing a given pattern of bits in plaintext with a different pattern of bits in ciphertext. Diffusion refers to transposing bits, that is changing their position in a systematic manner. Applying both techniques increase the difficulty of detecting useful patterns in ciphertext.

Testing for the presence of low-entropy encrypted data may be difficult for IT and Cybersecurity staff. Applications are available to measure ciphertext entropy, however the knowledge and skills required for their use is not part the NIST NICE framework nor covered by the COMPTIA Security+ Exam (Newhouse, 2017). The next section of this paper details how the proposed method, data compression, may be used as a proxy measurement for entropy.

A literature search was performed to find related work. The paper Relationship Between Entropy and Test Data Compression (Balakrishnan & Touba, 2007), examines the performance of different compression techniques and for test data generated by system-on-a-chip designs. Entropy measurements are used to establish theoretical limits for the amount of compression. The authors' detailed examination of compression algorithms is useful for those considering similar problems.

On Compression of Data Encrypted with Block Ciphers (Klinc et al., 2009) investigates the compression of high-entropy ciphertext. The authors discuss an approach may provide significant compression for certain sets of ciphertext inputs.

Distinguishing Compressed and Encrypted File Fragments (De Gaspari et al., 2020) examines the problem of using entropy to detect encrypted files when compressed files may have similar entropy values. Current approaches were not successful and the authors created a learning-based classifier, ExCoD that can differentiate between the two types of files.

COMPRESSION AS A PROXY FOR ENTROPY

Data compression is designed to decrease the amount of storage required for a given set of data by finding patterns that can be reduced to more compact representations. This led to the construction of a hypothesis for this paper and a supporting lemma:

- The author hypothesizes that it is possible to use compressibility as a proxy measurement of entropy for detecting the use of Electronic Code Book (ECB) mode encryption with a modern cryptographic algorithm and selected plaintext consisting of a single repeated character.
- The author proposes that there is mathematical relationship between the measured entropy of encrypted data and the measured amount of compression provided by readily-available applications.

An experiment was designed to generate data to test the hypothesis and the associated lemma. First, a series of plaintext files were specified and then generated. Then a process was designed to:

- Encrypt the plaintext files.
- Measure the entropy of the encrypted file.
- Compress the encrypted file.
- Calculate the change in size of the compressed file vs the uncompressed file.

A file size of 2,560 bytes was chosen. This will contain multiple blocks of data from the largest block size in use among modern cryptosystems. Plaintext files containing repeated single characters, nulls (0x00), were generated. Plaintext files containing a single set and multiple repeated sets of random binary data were also generated. The Linux “dd” utility was used to read random data from the Linux /dev/random device.

There are two types of file compression in general use, lossless and lossy. Lossless compression has the ability to exactly recreate the original data and is widely used for documents and other files that will suffer from changes to the data. Lossy compression provides a greater amount of compression at the cost of an inexact replication of the original data. Lossy compression is useful for video, images, audio, and other data where small differences are acceptable. Lossless compression was selected for use to allow for an exact recreation of the original files.

Two commonly available lossless algorithms are bzip2 and zip, both of which can be used from the command line for automating the encryption, compression and measurement process (“the process”) on Windows and Linux. Experiments found that gzip encryption provides a greater amount of compression for ciphertext files and gzip was selected for use.

Data was encrypted using the “openssl” command-line encryption application. It supports many modern encryption algorithms and has command line options that assist with automating the process.

Entropy was measured using the Linux “ent” command-line application. The accuracy was measured against the Cryptool Window GUI application used for cryptography research and education. Results from the tools were compared and there was less than a 2% difference in measured entropy values. The “ent” application was selected for its ability to be used from the command line to automate the process.

The experiments and resulting data are described Experiments and Data below. Conclusions and Further Research describes the application of these techniques to Cybersecurity & Information Assurance education.

EXPERIMENTS AND DATA

Testing was performed using Kali Linux version 2021.2 in a VMWare virtual machine. Encryption was performed using the openssl utility version 1.1.1k. Entropy was measured using the ent utility, build date 11/22/20. Encryption, compression, and entropy measurements were automated with zsh shell scripts.

Input files were constructed to address the hypothesis and the associated lemma. A 2,560 byte file consisting of nulls (0x00, all bits set to "0") was used as plaintext. The use of a single 8-bit character negates concerns about byte-alignment within each block of data. Each algorithm and mode were tested with a key of all nulls, 3 randomly generated keys, and a key with all bits set to 0xFF repeated (all bits set to "1"). For those modes requiring Initialization Vectors (IVs), each algorithm and mode were tested with an IV containing all nulls, 3 random IVs, and an IV containing 0xFFs.

Three encryption algorithms and three modes of encryption were selected for testing. The first algorithm is the Data Encryption Standard (DES) which has a 56-bit key and a 64-bit block size. DES was originally developed by IBM and is the oldest of the three algorithms. The second algorithm is the SEED algorithm which has a 128 bit-key and a 128-bit block size. SEED was developed by the South Korean government for use by South Korean government, defense, and commercial organizations. The third algorithm, the Advanced Encryption Standard (AES), is a modern symmetric cipher endorsed by the United States' National Institute of Standards and Technology (NIST). AES can use a 128-bit, 192-bit, or 256-bit key and has a 128-bit block size. The 256-bit key size was used for these tests.

Each algorithm was tested in its Electronic Code Book (ECB), Cipher Block Chaining (CBC), and Output Feed Back (OFB) modes. For any given algorithm and key, ECB provides the same result each time a given piece of data is encrypted using that key. CBC and OFB are more secure modes that use an initialization vector (IV) in addition to the key.

Plaintext files containing sequences of random characters were included to measure their effect on entropy and compression. The first file contained 2,560 bytes of random characters (0x00-0xFF), the second contained 5 repeated sections of 512 random bytes, the third contained 10 repeated sections of 256 random bytes, and the fourth contained 20 repeated sections of 128 random bytes.

During testing it was noted that the size of the ciphertext filename had a small effect on the resulting file size due to large filenames requiring more storage space than small filenames. The calculations for compression percentages made accommodations for this issue.

The input files, shell scripts used to automate the process, and files containing experimental results are have been uploaded to Github and are available to the public.

Proving the Hypothesis

Figure 1 below shows the compressed size of the ciphertext output for the ECB algorithm as a percentage of its original size, the type of plaintext, and the encryption key. In all cases encryption of plaintext consisting of nulls resulted in ciphertext that was compressed to a small fraction (2%-3%) of its original file size.

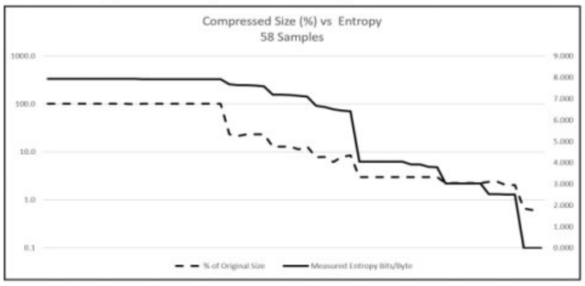


Figure 1 Compressed Size of Ciphertext vs

Measured Entropy

Note - (R) =Random				Compressed
Algorithm	Plaintext	Key	Entropy	Size
DESECB	Nulls	Key 0 - Nulls	3.030	2.26%
DESECB	Nulls	Key 1 - (R)	3.027	2.26%
DESECB	Nulls	Key 2 - (R)	3.030	2.26%
DESECB	Nulls	Key 3 - (R)	3.030	2.26%
DESECB	Nulls	Key 4 - 0xff's	3.030	2.26%
DESECB	Random (R)	Key 0 - Nulls	7.942	101.25%
DESECB	5x512 (R)	Key 0 - Nulls	7.637	23.44%
DESECB	10x256 (R)	Key 0 - Nulls	7.187	12.93%
DESECB	20x128 (R)	Key 0 - Nulls	6.678	7.75%
DESECB	Random (R)	Key 1 - (R)	7.933	101.25%
DESECB	5x512 (R)	Key 1 - (R)	7.577	23.40%
DESECB	10x256 (R)	Key 1 - (R)	7.106	12.97%
DESECB	20x128 (R)	Key 1 - (R)	6.406	8.57%
SEED128ECB	Nulls	Key 0 - Nulls	4.054	3.03%
SEED128ECB	Nulls	Key 1 - (R)	4.050	3.03%
SEED128ECB	Nulls	Key 2 - (R)	3.930	3.03%
SEED128ECB	Nulls	Key 3 - (R)	3.800	3.03%
SEED128ECB	Nulls	Key 4 - 0xff's	4.050	3.03%
SEED128ECB	Random (R)	Key 0 - Nulls	7.922	101.32%
SEED128ECB	5x512 (R)	Key 0 - Nulls	7.670	23.56%
SEED128ECB	10x256 (R)	Key 0 - Nulls	7.188	13.12%
SEED128ECB	20x128 (R)	Key 0 - Nulls	6.613	7.92%
AES256ECB	Nulls	Key 0 - Nulls	3.803	2.99%
AES256ECB	Nulls	Key 1 - (R)	4.047	2.99%
AES256ECB	Nulls	Key 2 - (R)	4.048	2.99%
AES256ECB	Nulls	Key 3 - (R)	4.054	2.99%
AES256ECB	Nulls	Key 4 - Nulls	3.923	2.99%
AES256ECB	Random (R)	Key 0 - Nulls	7.926	101.28%
AES256ECB	5x512 (R)	Key 0 - Nulls	7.623	23.45%
AES256ECB	10x256 (R)	Key 0 - Nulls	7.168	13.04%
AES256ECB	20x128 (R)	Key 0 - Nulls	6.453	7.88%

Figure 2 – ECB Compression Results

Encrypting a totally random plaintext file results in a file that is a approximately 1% larger than the original file for all algorithms. The larger size is due to gzip compression's internal representation of data that has few useful patterns in the ciphertext; the original file is replicated with additional overhead needed by the gzip algorithm. The table also shows that compressing files with repeated random sequences resulted in significant decreases in size and lower values for entropy.

The results shown in Figure 1 and 2 above prove the Hypothesis. Lossless compression can be used as a proxy for entropy to detect the use of ECB using chosen plaintext for the three algorithms tested.

Proving the Lemma

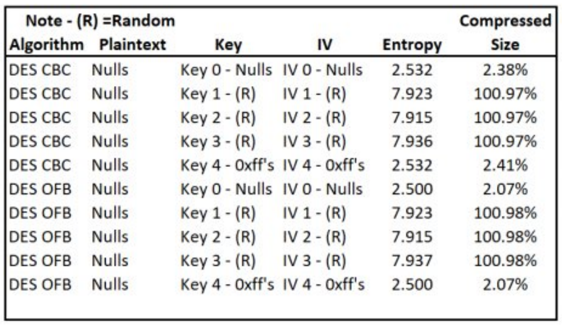
As shown in figures 1 and 2 above, compression and entropy were measured for 58 ciphertext and plaintext files. Entropy measurements ranged from 0.00 up to 7.947 where 0.00 is the minimum possible and 8.00 is the maximum possible. The compressed files ranged from 0.6% to 101.2% of the uncompressed file size.

The graph in Figure 1 shows that a logarithmic relationship exists between measured entropy (0-8) and the compressed file size expressed as a percentage of its original size, proving the lemma.

Detecting Other Vulnerabilities, DES

"Weak Keys"

Some implementations of DES are known to have issues with certain keys, such as repeated 0x00 (nulls, no bits set) and repeated 0xff (all bits set). These resulted in highly compressed files (2% of original size) and they had lower entropy than files created with random keys (2.5 vs 7.9) as shown in Figure 3 below.



Note - (R) =Random					Compressed
Algorithm	Plaintext	Key	IV	Entropy	Size
DES CBC	Nulls	Key 0 - Nulls	IV 0 - Nulls	2.532	2.38%
DES CBC	Nulls	Key 1 - (R)	IV 1 - (R)	7.923	100.97%
DES CBC	Nulls	Key 2 - (R)	IV 2 - (R)	7.915	100.97%
DES CBC	Nulls	Key 3 - (R)	IV 3 - (R)	7.936	100.97%
DES CBC	Nulls	Key 4 - 0xff's	IV 4 - 0xff's	2.532	2.41%
DES OFB	Nulls	Key 0 - Nulls	IV 0 - Nulls	2.500	2.07%
DES OFB	Nulls	Key 1 - (R)	IV 1 - (R)	7.923	100.98%
DES OFB	Nulls	Key 2 - (R)	IV 2 - (R)	7.915	100.98%
DES OFB	Nulls	Key 3 - (R)	IV 3 - (R)	7.937	100.98%
DES OFB	Nulls	Key 4 - 0xff's	IV 4 - 0xff's	2.500	2.07%

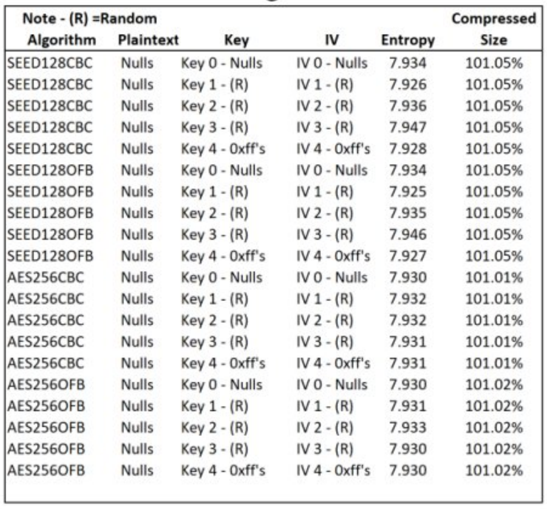
Figure 3 - Key and IV Issues with

OPENSSL DES Encryption

These results were unexpected by the researcher. Further investigation showed that this vulnerability and the underlying implementation issues are well known.

SEED and AES – Non-ECB Modes

The SEED and AES encryption algorithms had a compressed file size slightly larger than the ciphertext files themselves, indicating little or no compression was possible when used the CBC and OFB modes. The output files had a consistently high entropy in excess of 7.92 as shown in Figure 4 below.



Note - (R) =Random					Compressed
Algorithm	Plaintext	Key	IV	Entropy	Size
SEED128CBC	Nulls	Key 0 - Nulls	IV 0 - Nulls	7.934	101.05%
SEED128CBC	Nulls	Key 1 - (R)	IV 1 - (R)	7.926	101.05%
SEED128CBC	Nulls	Key 2 - (R)	IV 2 - (R)	7.936	101.05%
SEED128CBC	Nulls	Key 3 - (R)	IV 3 - (R)	7.947	101.05%
SEED128CBC	Nulls	Key 4 - 0xff's	IV 4 - 0xff's	7.928	101.05%
SEED128OFB	Nulls	Key 0 - Nulls	IV 0 - Nulls	7.934	101.05%
SEED128OFB	Nulls	Key 1 - (R)	IV 1 - (R)	7.925	101.05%
SEED128OFB	Nulls	Key 2 - (R)	IV 2 - (R)	7.935	101.05%
SEED128OFB	Nulls	Key 3 - (R)	IV 3 - (R)	7.946	101.05%
SEED128OFB	Nulls	Key 4 - 0xff's	IV 4 - 0xff's	7.927	101.05%
AES256CBC	Nulls	Key 0 - Nulls	IV 0 - Nulls	7.930	101.01%
AES256CBC	Nulls	Key 1 - (R)	IV 1 - (R)	7.932	101.01%
AES256CBC	Nulls	Key 2 - (R)	IV 2 - (R)	7.932	101.01%
AES256CBC	Nulls	Key 3 - (R)	IV 3 - (R)	7.931	101.01%
AES256CBC	Nulls	Key 4 - 0xff's	IV 4 - 0xff's	7.931	101.01%
AES256OFB	Nulls	Key 0 - Nulls	IV 0 - Nulls	7.930	101.02%
AES256OFB	Nulls	Key 1 - (R)	IV 1 - (R)	7.931	101.02%
AES256OFB	Nulls	Key 2 - (R)	IV 2 - (R)	7.933	101.02%
AES256OFB	Nulls	Key 3 - (R)	IV 3 - (R)	7.930	101.02%
AES256OFB	Nulls	Key 4 - 0xff's	IV 4 - 0xff's	7.930	101.02%

Figure 4 High-Entropy – SEED and AES in

CBC and OFB Modes

Entropy measurements and the size of compressed files indicate that these algorithms successfully transformed plaintext with very low entropy, 0.0, into very high entropy ciphertext. As shown in Figure 4, a possible indicator of high-entropy ciphertext is an expansion in size when compressed with GNU Zip.

OTHER APPLICATIONS

The use of compression as a proxy measurement for entropy would be useful for educating professionals in the workplace and students through hands-on methodologies such as Discovery Learning and Experiential Learning. Exercises based on compressed file size may be used to provide competency at Levels 1, 2 and 3 of Bloom's Revised Taxonomy (Remembering, Understanding, Applying) (Krathwohl, 2002, p. 215). Comparisons can be made using easily understood metrics such as file sizes pre- and post-compression. Identification of repeated sequences can be related to compression, where repeated sequences are stored once and compact "pointers" act as placeholders for the sequences in the compressed file.

Jerome Bruner's Theory of Discovery proposes that learners use past experiences and knowledge to discover new facts and relationships through hands-on interaction with their environment (Mcleod, 1970). Experiential Learning also emphasizes hands-on experimentation. Lab experiments that provide for rapid feedback from experimentation will help students quickly build an internal representation for the relation between compressibility and entropy. Per Kolb, "the methods of grasping experience are abstract conceptualization and concrete experience "(Kolb, 1984)(Cherry, 2020). Conventional curriculums based on mathematics is focused on abstract representations. A complementary or alternate approach can use hands-on experience with the learner performing tasks and observing results in real-time to build their own base of knowledge and experience. Students will then be better prepared to understand abstract concepts such as entropy and Shannon's work on Information Theory.

CONCLUSIONS AND FURTHER RESEARCH

Conclusions

The experimental results support the conclusion that ECB-mode encryption for symmetric algorithms such as DES, SEED, and AES can be detected via use of chosen plaintext input and then compressing the ciphertext output with the GNU Zip lossless compression application. This conclusion may be generalized due to ECB-mode's inherently vulnerability to this type of analysis, and because a number of file compression algorithms and applications would be able to efficiently compress this type of data.

The experimental results indicate there is a logarithmic relationship between the compressed file size and the measured entropy of ciphertext files encrypted using the selected modern symmetric-key algorithms, proving the lemma. This relationship was also noted in the plaintext files used for encryption.

An unexpected finding is the potential to identify the encryption algorithm (DES, SEED, AES, etc.) using chosen ciphertext and the amount of compression. Small and consistent differences are noted between different algorithms and encryption modes. Non-ECB modes for s returned a consistent size of 101.98% compressed, SEED 101.05% compressed, and AES 101.01% compressed. An opportunity for further research would test other algorithms and key sizes (Triple-DES, AES-128, Blowfish, etc.).

Further Research

Additional research opportunities include determining if transmitted data has byte-alignment or encoding issues that make detection of ECB more difficult when data is distributed across packets for transmission. It is also possible that lossy compression may be useful for finding patterns that are not detected with lossless compression. A comparison between the two types of compression would be useful.

ACKNOWLEDGEMENTS

Capitol Technology University was kind enough to allow the use of its campus and facilities for this research. The late Dr. Win Wenger provided valuable guidance on the works of Drs. Piaget, Bruner & Kolb. Dr. William Butler, Dr. Sandy Antunes and Professor Frank Davis provided much helpful input and encouragement.

REFERENCES

Bai, W., Pearson, M., Kelley, P. G., & Mazurek, M. L. M. L. (2020, October 22). *Improving non-experts' understanding of end-to-end encryption*. EUSEC20. Retrieved November 20, 2021, from https://eusec20.cs.uchicago.edu/eusec20-Bai.pdf

Balakrishnan, K. J., & Touba, N. A. (2007). Relationship between entropy and Test Data Compression. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, *26*(2), 386–395.

https://doi.org/10.1109/tcad.2006.882600

Cherry, K. (2020, May 15). *The David Kolb theory of how experience influences learning*. Verywell Mind. Retrieved March 22, 2022, from https://www.verywellmind.com/experiential-learning-2795154

De Gaspari, F., Pagnotta, D. H. G., De Carli, L., & Mancini, L. V. (2020, October 15). *Encod: Distinguishing compressed and encrypted File Fragments*. arxiv.org. Retrieved October 13, 2021, from https://arxiv.org/pdf/2010.07754.pdf

Klinc, D., Hazay, C., Jagmohan, A., Krawczyk, H., & Rabin, T. (2009). On compression of data encrypted with block ciphers. *2009 Data Compression Conference*. https://doi.org/10.1109/dcc.2009.71

Kolb, D. A., & Kolb. (1984). *Experiential learning: Experience as the source of learning and development, 2nd edition*. Pearson. Retrieved March 20, 2022, from https://www.pearson.com/us/higher-education/program/Kolb-Experiential-Learning-Experience-as-the-Source-of-Learning-and-Development-2nd-Edition/PGM183903.html

Krathwohl, D. R. (2002). A revision of Bloom's Taxonomy: An Overview. *Theory Into Practice*, *41*(4), 212–218. https://doi.org/10.1207/s15430421tip4104_2

Marczak, B., & Scott-Railton, J. (2021, June 29). *Move fast and roll your own crypto: A quick look at the confidentiality of zoom meetings*. The Citizen Lab. Retrieved December 8, 2021, from https://citizenlab.ca/2020/04/move-fast-roll-your-own-crypto-a-quick-look-at-the-confidentiality-of-zoom-meetings/

Mcleod, S. (1970, January 1). *[bruner - learning theory in education]*. Simply Psychology. Retrieved June 9, 2021, from https://www.simplypsychology.org/bruner.html

Newhouse, W., Keith, S., & Scribner, B. (2017, August). National Initiative for Cybersecurity Edcation NIST 800-181, KSA K0274

Shannon, C. (n.d.). *A Mathematical Theory of Cryptography*. International Association for Cryptologic Research. Retrieved October 10, 2021, from https://www.iacr.org/museum/shannon/shannon45.pdf