Critical Design Review

Death Star Image Exfiltration

Prepared by:

**Group 8**

Chase Ennis [ennis.18@wright.edu](mailto:ennis.18@wright.edu)

Mason McDaniel [mcdaniel.121@wright.edu](mailto:mcdaniel.121@wright.edu)

Nicholas Michael [michael.102@wright.edu](mailto:michael.102@wright.edu)

Michael Mowad [mowad.3@wright.edu](mailto:mowad.3@wright.edu)

Cade Wrinkle [wrinkle.2@wright.edu](mailto:wrinkle.2@wright.edu)

Sponsored by:

Dr. Michael Raymer [michael.raymer@wright.edu](mailto:michael.raymer@wright.edu)

Dr. Weisong Wang [weisong.wang@wright.edu](mailto:weisong.wang@wright.edu)

Document Tracking Information:

Document Version: 1.0

Date: 11/22/24

Final Approved Date:

Final Approved By:

Table of Contents:

[1. Executive Summary 3](#_Toc1599046855)

[1.a Purpose of this Document 4](#_Toc638492156)

[1.b Design Scope 4](#_Toc1724805271)

[1.c Intended Audience and Document Overview 4](#_Toc1396877181)

[1.d Definitions, Acronyms, and Abbreviations 4](#_Toc133320655)

[1.e Document Conventions 5](#_Toc1653633799)

[1.f References and Acknowledgements 5](#_Toc623360368)

[2. Problem Statement 15](#_Toc1692432892)

[2.a Historical Introduction 16](#_Toc814431792)

[2.b Alternative Approaches 16](#_Toc323446772)

[2.c Proposed Solution 20](#_Toc512119965)

[2.d Impact of Success 21](#_Toc742141301)

[3. Context of Design Solution 22](#_Toc1713433627)

[3.a Design Objectives 22](#_Toc90229717)

[3.b Design Assumptions 23](#_Toc1072596038)

[3.c Design Requirements 23](#_Toc1057456809)

[3.d Design Constraints 24](#_Toc1098372619)

[3.e Design Standards 24](#_Toc1851947365)

[3.f Design Functionality 25](#_Toc752801786)

[3.g User Characteristics 25](#_Toc23541703)

[3.h Operating Environment 25](#_Toc2008648697)

[4. Technical Approach 26](#_Toc793071434)

[4.a Hardware/Software Tradeoffs 27](#_Toc772852041)

[4.c Hardware Design 30](#_Toc202406011)

[5. Appendix: 40](#_Toc1194892526)

[5.a Gant Chart 41](#_Toc24494924)

[5.b Budget 41](#_Toc334898129)

[5.c Requirements Testing 41](#_Toc705469792)

# 1. Executive Summary

This document describes the proposed exfiltration solution hereafter referred to as the Death Star Image Exfiltration Scheme (DSIES). DSIES seeks to discriminate between Death Star and non-Death Star images, transmit the 10 Death Star images wirelessly over an encrypted channel within a ten-minute interval, verify the 10 Death Star images using MD5 checksums, identify the weaknesses highlighted in the Death Star Images and display them within a mobile app. It shall describe the project scope, its intended use, its specifications, how it is to be tested and verified, and by what metrics will success be evaluated.

# 1.a Purpose of this Document

This document describes the construction, purpose, and functionality of the Death Star Image Exfiltration Scheme. The hardware system architecture consists of a Raspberry Pi 4 equipped with an NRF52840-DONGLE transceiver device, which transmits data across a glass barrier to a Linux PC equipped with a corresponding NRF52840-DONGLE. All hardware components are sourced from reputable vendors. The software architecture includes an image recognition system that identifies specific images of the Death Star, encrypted data transmission using OpenSSL AES-128, and a mobile application that receives and displays the processed images. The server-side application crops images to isolate marked weaknesses before transmitting them to the mobile app.

# 1.b Design Scope

The Death Star Image Exfiltration Scheme is designed to operate within a controlled, line-of-sight environment, utilizing image recognition and encrypted transmission capabilities. The primary focus is to identify, isolate, and securely transmit images containing specific Death Star characteristics (e.g., red-circled weaknesses) from a Raspberry Pi device to a Linux PC through a glass barrier. The Raspberry Pi captures and processes 1024x1024 images, with specialized software to identify Death Star images based on predefined characteristics.

# 1.c Intended Audience and Document Overview

This document is intended for project professors, Wright State University staff, and technical writers. All readers should refer to Section 1 for a summary of the project. Readers  
interested in using the system should read at least Section 2 and Section 3.6. For a full  
picture of the system, readers should view the entire document.

# 1.d Definitions, Acronyms, and Abbreviations

**AES** – Advanced Encryption Standards

**ESB** – Enhanced Shock Burst Protocol

**PNG** – Portable Network Graphic

**IEEE** – Institute of Electrical and Electronics Engineers

**RF** – Radio Frequency

# 1.e Document Conventions

Citations are in IEEE format.

# 1.f References and Acknowledgements

**Wireless Transmission Protocol References**

[1] “Enhanced Shockburst.” Nordic semiconductor. [Online]. Available: <https://docs.nordicsemi.com/bundle/ncs-latest/page/nrf/protocols/esb/index.html>. Accessed: October 14, 2024.

[2] “nRF52840 Product Specification.” Nordic semiconductor. [Online]. Available: <https://docs.nordicsemi.com/bundle/ps_nrf52840/page/keyfeatures_html5.html> Accessed: October 14, 2024.

[3] “Nordic Semiconductor nRF52840-DK.” Mouser Electronics. [Online]. Available:

<https://mou.sr/40VS31i>. Accessed: October 14, 2024.

[4] *“*DIGI XBEE 3 ZIGBEE MESH KIT.” Digikey. [Online]. Available: [https://mm.digikey.com/Volume0/opasdata/d220001/medias/docus/6346/XK3-Z8S- WZM.pdf](https://mm.digikey.com/Volume0/opasdata/d220001/medias/docus/6346/XK3-Z8S-WZM.pdf). Accessed: October 14, 2024.

[5] “How to Transform a RaspberryPi Into a Universal Zigbee and Z-Wave Bridge.” Hackernoon. [Online]. Available: [https://hackernoon.com/how-to-transform-a-raspberrypi- into-a-universal-zigbee-and-z-wave-bridge-xy1ay3ymz](https://hackernoon.com/how-to-transform-a-raspberrypi-into-a-universal-zigbee-and-z-wave-bridge-xy1ay3ymz). Accessed: October 14, 2024.

[6] “Design Considerations for High-Throughput SDR Systems.” ElectronicDesign. [Online]. Available: [https://www.electronicdesign.com/markets/automation/article/21157953/per-vices- corp-design-considerations-for-high-throughput-sdr-systems](https://www.electronicdesign.com/markets/automation/article/21157953/per-vices-corp-design-considerations-for-high-throughput-sdr-systems) Accessed: October 16, 2024

[7] “Radioberry.” GitHub Repository. [Online]. Available: [https://github.com/pa3gsb/ Radioberry-2.x/wiki](https://github.com/pa3gsb/Radioberry-2.x/wiki). Accessed: October 14, 2024.

[8] “AURSINC SDR Radioberry V2.0 for Raspberry Pi 4/4B, Radio Card (Raspberry Pi 4 Hat) Analog Devices AD9866 & 10CL025 12-bit Broadband Modem for Ham Radio SDR Transceiver Software Defined Radio” Amazon.com. [Online]. Available: <https://a.co/d/2s4OAS2>. Accessed: Oct. 16, 2024.

[9] Rust, Evan. “Scan QR Codes in Real-Time with Raspberry Pi.” Hackster.io. [Online]. Available: [https://www.hackster.io/gatoninja236/scan-qr-codes-in-real-time-with-Raspberry-Pi-a5268b](https://www.hackster.io/gatoninja236/scan-qr-codes-in-real-time-with-raspberry-pi-a5268b). Accessed: Oct. 16, 2024.

[10] Tripathi, Pankaj. “How Much Data Can QR Code Hold?” UNIQODE. Available: <https://www.uniqode.com/blog/qr-code/how-much-data-can-qr-code-hold>. Accessed: Oct. 16, 2024

[11] Suvak, Dave. “IrDA and Bluetooth: A Complementary Comparison.” Extended Systems. [Online]. Available: <http://alumni.cs.ucr.edu/~csyiazti/courses/cs260/downloads/IrDA_vs_Bluetooth.pdf>. Accessed: Oct. 14, 2024.

[12] “USB IrDA Kit.” IRDroid. [Online]. Available: <https://irdroid.eu/product/usb-irda-kit/>. Accessed: Oct. 16, 2024.

[13] “Unit 5: IrDA Communications Protocol.” Digilent. [Online]. Available: <https://digilent.com/reference/microprocessor/basys-mx3/unit-5/start>. Accessed: Oct. 16, 2024.

**Wireless Transceiver Hardware References**

[1] "Makerfire Arduino NRF24L01 Wireless Transceiver Module," Amazon, [Online]. Available: <https://www.amazon.com/Makerfire-Arduino-NRF24L01-Wireless-Transceiver/dp/B00O9O868G>. Accessed: October 11, 2024.

[2] “nRF24L01+ Preliminary Product Specification," SparkFun, Jan. 2020. [Online]. Available: <https://www.sparkfun.com/datasheets/Components/SMD/nRF24L01Pluss_Preliminary_Product_Specification_v1_0.pdf>. Accessed: October 11, 2024.

[3] "NRF24L01 Tutorial: Arduino Wireless Communication," Instructables. [Online]. Available: <https://www.instructables.com/NRF24L01-Tutorial-Arduino-Wireless-Communication/>. Accessed: October 11, 2024.

[4] "Adafruit NRF24L01+ 2.4GHz RF Transceiver - 2Mbps, 100mW" Digi-Key.[Online]. Available: <https://www.digikey.com/en/products/detail/adafruit-industries-llc/4062/9843410?s=N4IgjCBcoLQBxVAYygMwIYBsDOBTANCAPZQDa4ArAEwIC6AvvYVWSACwAMAbCw0A>. Accessed: October 11, 2024.

[5] *“*nRF52840*”* Mouser. [Online]. Available: <https://www.mouser.com/datasheet/2/297/nrf52840_soc_v3_0-2942478.pdf>.Accessed: October 14, 2024.

[6] “NRF52840-DONGLE.” Digikey. [Online]. Available: <https://www.digikey.com/short/72z130hw>. Accessed: October 21, 2024.

[7] “nRF52840 Dongle.” Nordic Semiconductor. [Online]. Available: <https://docs.nordicsemi.com/bundle/ug_nrf52840_dongle/page/UG/nrf52840_Dongle/intro.html>. Accessed: October 21, 2024.

[8] “Approval Sheet” Raytac Corporation. [Online]. Available: <https://cdn.sparkfun.com/assets/learn_tutorials/8/2/0/Raytac_MDBT50Q.pdf.> Accessed: October 21, 2024.

[9] “SparkFun Pro nRF52840 Mini - Bluetooth Development Board” SparkFun. [Online]. Available: <https://www.sparkfun.com/products/15025>. Accessed: October 21, 2024.

[10] “Nordic Thingy:91” Nordic Semiconductors. [Online]. Available: <https://www.nordicsemi.com/-/media/Software-and-other-downloads/Product-Briefs/Nordic-Thingy-91-PB-v2.1.pdf>. Accessed: October 18, 2024.

[11] “NRF6943“ Digikey. [Online]. Available: <https://www.digikey.com/en/products/detail/nordic-semiconductor-asa/NRF6943/10291811>. Accessed: October 18, 2024.

**Image Recognition Algorithm References**

[1] A. Satya, J. Prakash, and P. Sriramya, “Accuracy Analysis for Image Classification and Identification of Nutritional Values Using Convolutional Neural Networks in Comparison with Logistic Regression Model,” *Journal of Pharmaceutical Negative Results*, vol. 13, no. SO4, doi: 10.47750/pnr.2022.13.s04.067. Available: <https://www.pnrjournal.com/index.php/home/article/view/951> Accessed: Oct 18, 2024.

[2] A. Thanda, “What is Logistic Regression? A Beginner's Guide [2023],” *CareerFoundry*, May 11, 2023. Available: <https://careerfoundry.com/en/blog/data-analytics/what-is-logistic-regression/>. Accessed: October 18, 2024.

[3] D. Cai, X. He and J. Han, "Training Linear Discriminant Analysis in Linear Time," *2008 IEEE 24th International Conference on Data Engineering*, Cancun, Mexico, 2008, pp. 209-217, doi: 10.1109/ICDE.2008.4497429. Available: <https://ieeexplore.ieee.org/document/4497429>. Accessed: Oct 18, 2024.

[4] D. Yates and M. Z. Islam, “FastForest: Increasing random forest processing speed while maintaining accuracy,” *Information Sciences*, vol. 557, pp. 130–152, Dec. 2020, doi: 10.1016/j.ins.2020.12.067. Available: <https://www.sciencedirect.com/science/article/pii/S0020025520312330>. Accessed: Oct. 18, 2024.

[5] E. Kavlakoglu, “Implementing linear discriminant analysis (LDA) in Python,” *IBM Developer*, Mar. 18, 2024. Available: <https://developer.ibm.com/tutorials/awb-implementing-linear-discriminant-analysis-python/>. Accessed: Oct. 18, 2024.

[6] “Linear Discriminant analysis in machine learning,” *GeeksforGeeks*, Mar. 20, 2024. Available: [https://www.geeksforgeeks.org/ml-linear-discriminant-analysis/#advanatages-disadvantages-of-using-lda](https://www.geeksforgeeks.org/ml-linear-discriminant-analysis/). Accessed: Oct. 18, 2024.

[7] N. Beheshti, “Random Forest Classification - towards data science,” *Medium*, Jan. 29, 2022. [Online]. Available: <https://towardsdatascience.com/random-forest-classification-678e551462f5>. Accessed: Oct 18, 2024.

[8] “What is Logistic Regression? - Logistic Regression Model Explained - AWS,” *Amazon Web Services, Inc.* Available: <https://aws.amazon.com/what-is/logistic-regression/>. Accessed: Oct. 18, 2024.

[9] “What is random forest?,” *IBM*. Available: <https://www.ibm.com/topics/random-forest>. Accessed: Oct. 18, 2024.

[10] S. Wiyono, D. S. Wibowo, and M. F. Hidayatullah, "Comparative Study of KNN, SVM, and Decision Tree Algorithm for Student’s Performance Prediction," *International Journal of Computing Science and Applied Mathematics*, vol. 8, no. 2, pp. 1–6, 2022.  
 Available: <https://iptek.its.ac.id/index.php/ijcsam/article/view/4360>. Accessed: Oct 18, 2024.

[11] P. T. Noi and M. Kappas, "Comparison of Random Forest, K-Nearest Neighbor, and Support Vector Machine Classifiers for Land Cover Classification Using Sentinel-2 Imagery," *MDPI Sensors*, vol. 18, no. 1, pp. 18-26, 2018.  
 Available: <https://www.mdpi.com/1424-8220/18/1/18>. Accessed: Oct. 18, 2024.

[12] L. A. Demidova, "Two-Stage Hybrid Data Classifiers Based on SVM and kNN Algorithms," *MDPI Symmetry*, vol. 13, no. 4, pp. 615–630, 2021.  
 Available: <https://www.mdpi.com/2073-8994/13/4/615>. Accessed: Oct. 18, 2024.

[13] J. B. Quinlan, "Induction of Decision Trees," *Machine Learning*, vol. 1, no. 1, pp. 81–106, 1986. Available: <https://link.springer.com/article/10.1023/A:1022643204877>. Accessed: Oct. 18, 2024.

[14] C. Cortes and V. Vapnik, "Support-Vector Networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995. Available: <https://link.springer.com/article/10.1007/BF00994018>. Accessed: Oct. 18, 2024.

[15] K. R. Varma and N. Krishnan, "KNN vs SVM: A Comparative Study," *GeeksforGeeks*, 2020. Available: <https://www.geeksforgeeks.org/svm-vs-knn-in-machine-learning/> Accessed: Oct. 18, 2024.

[16] A. Biswal, “Convolutional Neural Network Tutorial,” Simplilearn.com, Sep. 21, 2022. <https://www.simplilearn.com/tutorials/deep-learning-tutorial/convolutional-neural-network>

[17] A. Lavin and S. Gray, “Fast Algorithms for Convolutional Neural Networks.” Available: <https://openaccess.thecvf.com/content_cvpr_2016/papers/Lavin_Fast_Algorithms_for_CVPR_2016_paper.pdf>

‌

**Image Recognition Software References**

[1] OpenCV, "Introduction and Overview," [Online]. Available: <https://docs.opencv.org/4.x/>. Accessed: Oct. 14, 2024.

[2] OpenCV, "Platforms," [Online]. Available: <https://opencv.org/platforms/>. Accessed: Oct. 14, 2024.

[3] StackOverflow, "What are the minimum system requirements for executing a simple project in TensorFlow?" [Online]. Available: <https://stackoverflow.com/questions/43985250/what-are-the-minimum-system-requirements-for-executing-a-simple-project-in-tenso>. Accessed: Oct. 15, 2024.

[4] P. Reddie, "YOLO: Real-Time Object Detection Explained," [Online]. Available: <https://pjreddie.com/darknet/yolo/>. Accessed: Oct. 15, 2024.

[5] AWS, "AWS Rekognition Overview," [Online]. Available: <https://aws.amazon.com/rekognition/>. Accessed: Oct. 14, 2024.

[6] AWS, "AWS Rekognition Pricing Details," [Online]. Available: <https://aws.amazon.com/rekognition/pricing/>. Accessed: Oct. 14, 2024.

[7] OpenCV, "License," [Online]. Available: <https://opencv.org/license/>. Accessed: Oct. 14, 2024.

[8] TensorFlow, "Install Guide," [Online]. Available: <https://www.tensorflow.org/install>. Accessed: Oct. 16, 2024.

[9] A. B. Alexey, "YOLO on Darknet GitHub," [Online]. Available: <https://github.com/AlexeyAB/darknet>. Accessed: Oct. 15, 2024.

[10] PyTorch, "PyTorch License," [Online]. Available: <https://pytorch.org/get-started/locally/>. Accessed: Oct. 14, 2024.

[11] PyTorch, "PyTorch Documentation," [Online]. Available: <https://pytorch.org/docs/stable/index.html>. Accessed: Oct 15, 2024.

[12] OpenCV, "Laptop minimum system requirement for OpenCV," [Online]. Available: <https://answers.opencv.org/question/179923/laptop-minimum-system-requirement-for-opencv/>. Accessed: Oct. 16, 2024.

[13] Raspberry Pi Forum, "Laptop minimum system requirement for OpenCV," [Online]. Available: <https://forums.raspberrypi.com/viewtopic.php?t=259562>. Accessed: Oct. 16, 2024.

[14] PyTorch Forum, "How to know the exact GPU memory requirement for a certain model?" [Online]. Available: <https://discuss.pytorch.org/t/how-to-know-the-exact-gpu-memory-requirement-for-a-certain-model/125466>. Accessed: Oct. 16, 2024.

**Encryption Algorithm References**

[1] S. Sari, "DES vs 3DES vs Blowfish vs AES," *Baeldung*, Baeldung. [Online]. Available: <https://www.baeldung.com/cs/des-vs-3des-vs-blowfish-vs-aes>

[2] A. Nurgaliyev and H. Wang, "Comparative study of symmetric cryptographic algorithms," *2021 5th International Conference on Cryptography, Security and Privacy (CSP)*, 2021, pp. 30-35. doi: 10.1109/CSP51677.2021.9634966. [Online]. Available: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9634966>.

[3] "AES vs IDEA," *This vs That*. [Online]. Available: <https://thisvsthat.io/aes-vs-idea>.

[4] "IDEA - International Data Encryption Algorithm," *TutorialsPoint*, [Online]. Available: <https://www.tutorialspoint.com/cryptography/idea_algorithm.htm>.

[5] W. Riddle, G. Quan, S. Salerno, E. Mendez, V. Formicola, N. El-Naga, and M. El-Haded, "AESPIE: Raspberry Pi AES performance evaluation using image encryption in C and Python," *in Advances in Artificial Intelligence, Software and Systems Engineering, Springer, 2024*, pp. 391-402. doi: 10.1007/978-3-031-40447-4\_30. [Online]. Available: <https://link.springer.com/content/pdf/10.1007/978-3-031-40447-4_30.pdf>.

[6] E. Fernando, D. Agustin, M. Irsan, D. F. Murad, H. Rohayani, and D. Sujana, "Performance comparison of symmetries encryption algorithm AES and DES with Raspberry Pi," *2019 International Conference on Computer Science, Information Technology, and Electrical Engineering (ICOMITEE)*, 2019, pp. 54-58. doi: 10.1109/ICOMITEE.2019.8986122. [Online]. Available: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8986122>.

[7] "Python DES Implementation," *GitHub Repository*. [Online]. Available: <https://gist.github.com/BlackRabbit-github/2924939>

[8] "Encrypting data with the Blowfish algorithm," *Embedded.com*, Aspen Core Network. [Online]. Available: <https://www.embedded.com/encrypting-data-with-the-blowfish-algorithm/>.

[9] "Mastering IDEA Encryption Algorithm in Java: A Comprehensive Guide," *Coding Tech Room*, Code Room Tech. [Online]. Available: <https://codingtechroom.com/tutorial/java-mastering-idea-encryption-algorithm-in-java-a-comprehensive-guide>.

[10] I. Bull, "Tutorial: AES encryption and decryption with OpenSSL," EclipseSource, EclipseSource, Jan. 17, 2017. [Online]. Available: <https://eclipsesource.com/blogs/2017/01/17/tutorial-aes-encryption-and-decryption-with-openssl/>.

[11] "EVP\_aes - OpenSSL," OpenSSL, 2018. [Online]. Available: <https://docs.openssl.org/1.1.1/man3/EVP_aes/>.

[12] "DES - OpenSSL," OpenSSL, 2016. [Online]. Available: <https://docs.openssl.org/1.0.2/man3/des/>.

[13] "Triple ECB mode - OpenSSL," OpenSSL, 2016. [Online]. Available: [https://docs.openssl.org/1.0.2/man3/des\_modes/#triple-ecb-mode](https://docs.openssl.org/1.0.2/man3/des_modes/).

[14] "Blowfish - OpenSSL," OpenSSL, 2016. [Online]. Available: <https://docs.openssl.org/1.0.2/man3/blowfish/>.

[15] "EVP\_CIPHER-IDEA - OpenSSL," OpenSSL, 2021. [Online]. Available: [https://docs.openssl.org/3.0/man7/EVP\_CIPHER-IDEA/#algorithm-names](https://docs.openssl.org/3.0/man7/EVP_CIPHER-IDEA/).

**Encryption Libraries References**

[1] S. Sharma, "How to install OpenSSL on Ubuntu Server," *Learn Ubuntu*, Nov. 28, 2022. [Online]. Available: https://learnubuntu.com/install-openssl/. Accessed: Oct. 21, 2024.

[2] R. Van de Paar, "Raspberry Pi: How to install OpenSSL 1.0.2 on Raspberry Pi3? (2 Solutions!!)," *YouTube,* Dec. 19, 2020. [Online]. Available: <https://www.youtube.com/watch?v=WkXeHAwaLHg>. Accessed: Oct. 21, 2024.

[3] A. Khan, "How to Update OpenSSL on Raspberry Pi," *Linux Hint*, 2022. [Online]. Available: <https://linuxhint.com/update-open-ssl-Raspberry-Pi/>. Accessed: Oct. 21, 2024.

[4] "How to Install the Libcrypto Library on Ubuntu," *CommandFound*, [Online]. Available: <https://commandfound.com/post/libcryptolib-not-found-ubuntu/>. Accessed: Oct. 21, 2024.

[5] "pycrypto 2.6.1," *AWS*, Oct. 17, 2013. [Online]. Available: <https://pypi.org/project/pycrypto/>. Accessed: Oct. 21, 2024.

[6] "How To Install gnupg on Ubuntu 20.04," *Installati.one*, [Online]. Available: <https://installati.one/install-gnupg-ubuntu-20-04/>. Accessed: Oct. 21, 2024.

[7] "OpenSSL License," OpenSSL Wiki, Jul. 22, 2020. [Online]. Available: <https://wiki.openssl.org/index.php/License>. Accessed: Oct. 21, 2024.

[8] "Libgcrypt," *GnuPG*, [Online]. Available: <https://www.gnupg.org/software/libgcrypt/index.html>. Accessed: Oct. 21, 2024.

[9] "PyCryptodome License," *PyCryptodome*, [Online]. Available: <https://pycryptodome.readthedocs.io/en/latest/src/license.html>. Accessed: Oct. 21, 2024.

[10] "GNU General Public License," *GNU Operating System*, [Online]. Available: <https://www.gnu.org/licenses/gpl-3.0.html>. [Accessed: Oct. 21, 2024].

[11] "Encrypt & Decrypt Files With Password Using OpenSSL," *ShellHacks*, [Online]. Available: <https://www.shellhacks.com/encrypt-decrypt-file-password-openssl/>. Accessed: Oct. 21, 2024.

[12] "The Libgcrypt Library," *GnuPG*, Apr. 6, 2023. [Online]. Available: <https://www.gnupg.org/documentation/manuals/gcrypt/>. Accessed: Oct. 21, 2024.

[13] D. Damian, "Implementing AES-256 Encryption and Decryption with PyCrypto in Python 3," *DNM*, Oct. 16, 2023. [Online]. Available: <https://dnmtechs.com/implementing-aes-256-encryption-and-decryption-with-pycrypto-in-python-3/>. Accessed: Oct. 21, 2024.

[14] "Encrypting and decrypting documents," *The GNU Privacy Handbook*, [Online]. Available: <https://www.gnupg.org/gph/en/manual/x110.html>. Accessed: Oct. 21, 2024.

[15] "IEEE CCM — AES CCM mode encryption." *Nordic Semiconductor ASA*, Oct. 1, 2024. [Online]. Available: <https://docs.nordicsemi.com/bundle/ps_nrf52840/page/ccm.html> Accessed: Oct. 25, 2024.

[16] "Adafruit NRF24L01+ 2.4GHz RF Transceiver - 2Mbps, 100mW" *DigiKey*. [Online]. Available: <https://www.digikey.com/en/products/detail/adafruit-industries-llc/4062/9843410?s=N4IgjCBcoLQBxVAYygMwIYBsDOBTANCAPZQDa4ArAEwIC6AvvYVWSACwAMAbCw0A>. Accessed: Oct. 11, 2024.

# 2. Problem Statement

Dissemination of information is, at times, necessary to protect against tyrannies and preserve civil well-being. To this end, a solution for an operative to exfiltrate a moderate quantity of image data over an air gap within a short transmission window is required. The solution should be able to identify key targets and weaknesses marked by operatives within them. The operatives frequently operate in hostile environments with portions of the RF spectrum monitored for communication over common standards, namely Wi-Fi, Bluetooth, GSM, and CDMA. Bespoke or rare communication schemes are understood to be subject to observation and recording as well; therefore, to preserve operational security, encryption is necessary. Additionally, to preserve the integrity of the information in transit, the exfiltrated images need to be verified. Once the images are exfiltrated, they must be disseminated via a server with an associated mobile app that highlights the marked weaknesses present in the targets.

Providing quantifiers, the DSIES must satisfy the following criteria:

* A USB thumb drive containing 50 images will be provided to a Raspberry Pi, which must identify and extract exactly 10 images of the Death Star from this set.
* These selected images, formatted as 1024x1024 PNGs, must then be encrypted and transferred to a Rebel Alliance server without using Wi-Fi, Bluetooth, or cellular connections.
* During the transfer, each image’s MD5 hash must be verified to ensure accuracy and integrity. This entire extraction, encryption, transfer, and verification process must be completed within 600 seconds.
* Once on the server, an automated system will analyze the images to isolate and highlight any structural weaknesses, displaying the result in a mobile app.

# 2.a Historical Introduction

Following World War II, espionage became a major weapon in every nation's arsenal, and it is still prevalent to this day. The need for secure information transfers is of high importance in espionage. In the proposed scenario of the transmission of top-secret Death Star weakness schematics, secure and undetectable communication is of utmost importance

# 2.b Alternative Approaches

**Wireless Transmission**

ESB is a proprietary protocol built upon ANT/ANT+. It supports the bidirectional communication of packets with buffering, acknowledgement, and automatic retransmission of dropped packets. It is available on Nordic Semiconductor’s nRF24 and nRF5 series devices. The transmission occurs over the 2.4-Ghz ISM band.

Zigbee is a standard collection of communication protocols used for wireless data transmission. Zigbee’s primary purpose is to be used as an alternative to technologies such as Bluetooth or Wi-Fi for smaller scale implementations. Communication is done over digital radio most commonly configured to 2.4-GHz signal. As it is a smaller scale implementation, Zigbee has overall lower data rates and requires relatively close proximity.

Software defined radio (SDR) is an implementation of standard radio protocols in a software environment. A basic SDR system may consist of a computer and an analog-to-digital converter, preceded by some form of RF front end. The main reason we did not choose this option is due to the cost of most preexisting SDR systems, which would exceed our budget. Furthermore, as there is no predefined encryption method for SDR, it would be very difficult to introduce our own encryption into the data transmissions.

Another option considered for implementation is the usage of QR codes. A QR code is a visual representation of data that can be easily scanned by devices with a camera. Most commonly QR codes are implemented for the purpose of providing a link to a website. In the context of the proposed system, connection to the internet is not an option therefore QR codes containing web links will not work. Instead, the QR codes would represent an encoding of the raw binary data of the image being transmitted. This would likely require multiple QR codes due to size limitations.

Laser IRDA is another technology that could be implemented to wirelessly transmit data. IRDA stands for Infrared Data Association which refers to a data transmission technology standard utilizing infrared light. The frequencies of these invisible light waves are configured in order to represent data that is being transmitted. Data is transmitted through the use of an infrared transmitter and an optical receiver that interprets the signal being received. Due to the nature of this technology, a line of sight is needed between the transmitter and receiver.

**Wireless Transmission Design Trades**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Option**  **[Req]** | **Rated Data Throughput [90]** | **Cost**  **[C 40]** | **Native Encryption Support [80]** | **Existing Software Libraries / SDKs [Opt]** | **Existing Tutorials [Opt]** | **Follows WSU environment and safety regulations** |
| Enhanced Shockburst (ESB) | 2 Mbps and 1 Mbps [2] | $46.37 [3] | Yes [2] | Yes [1] | Yes [1] | Yes |
| Zigbee | 250 Kbps [4] | $100 for kit [4] | AES-128/256 [4] | Yes [4] | Yes [5] | Yes |
| Software Defined Radio | Up to 400 Gb/s [6] | $159.99 [8] | No [7] | Yes [7] | Yes [7] | Yes |
| QR Code | 180 Kbps \* [10] | $12.49 [10] | No [9] | Yes [9] | Yes [9] | Yes |
| Laser IRDA | 9600 bps – 16 Mbps [11] | 80€ [12] | No [11] | No | Yes [13] | No |

**Wireless Transmission Design Choice:**

To transmit at least 10 1024x1024 images over a 600 second interval, the data rate of the transmission should be more than 420 Kbps. QR codes were removed from consideration due to the low supported data rate. IrDA can operate with a wide range of data rates, some of which are satisfactory and others that are unsatisfactory; it is implementation dependent. IrDA would require more effort to implement the communication protocol atop the hardware layer than ESB and Zigbee, due to the lack of satisfactory libraries. For a single channel, the data rate of Zigbee is below the minimum viable rate. Additionally, the barrier cost of Zigbee is higher than ESB, so ESB is preferred over Zigbee.

**Image Recognition Design Trades**

Image recognition is the process of identifying objects, shapes, colors, patterns, and features within images. It depends on several factors such as speed, accuracy and performance. There are several major methods to use: K-Nearest Neighbor, Decision Tree, Support Vector Machine, Random Forest, Linear Discriminant Classifier, and Logistic Regression are several of the most notable methods.

K-Nearest Neighbor is a straightforward machine learning classifier that delivers class predictions for new data based on how similar it is to the "k" most similar points or neighbors in each training dataset. Decision Tree: The decision tree is a form of machine learning for image classification that makes decisions through a structured tree by evaluating a sequence of queries about specific features extracted from the image. The support vector machine is a supervised machine learning algorithm that classifies data by finding a hyperplane that separates data points of different classes. A machine learning algorithm that uses many decision trees to make predictions. Linear Discriminant Classifier is a classifier that finds a linear combination of features to separate two or more classes of data. Logistic regression is a supervised machine learning algorithm mainly used for binary classification tasks. Convolutional Neural Networks are often utilized for image classification because the process concerns the convolution of a kernel value over all pixels in an image in order to assign weights to each region of the image.

Each of these methods have benefits and tradeoffs in terms of speed, ease of use, data size limit, training time, and accuracy.

**Image Recognition Design Trades Table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Option**  **[Req]** | **Speed & Efficiency**  **[90]** | **Ease of Integration**  **[Opt]** | **Training Time**  **[Opt]** | **Accuracy & Precision**  **[60,120]** |
| K-Nearest Neighbor | Slow [10] | Easy [15] | No training required [15] | Moderate [10] |
| Decision Tree | Fast [10] | Easy [13] | Moderate [13] | High [10] |
| Support vector machine | Slow (Complex Kernels) [14] | Complex [14] | Long [14] | Very High [11] |
| Random Forest | Slow [9] | Complex [9] | Moderate [5] | High [7] |
| Linear Discriminant Classifier | Fast [6] | Easy [6] | Moderate [3] | High [5] |
| Logistic Regression | Fast [2] | Easy [8] | Fast [8] | Moderate [1] |
| Convolutional Neural Networks (CNN) | Fast [16] | Easy [17] | Fast [17] | High [16] |

**Image Recognition Design Explanation:**

The chart compares several Image Recognition methods including “K-Nearest Neighbor (KNN)”, “Decision Tree”, “Support Vector Machine”, “Random Forest”, “Linear Discriminant Classifier”, and “Logistic Regression”. They are rated based on Sample Size, Speed & Efficiency, Ease of Integration, Training Time, and Accuracy & Precision. If an option got a “poor” rating for Sample size, it means that it can only handle a small data set, and vice versa. If an option is “slow” for Speed & Efficiency, that means that the option is a time-consuming option, and vice versa. If an option is “complex” for Ease of Integration, that means that it may be difficult to intergrade into the project. If an option is “Long” for Training Time, that means that it takes a while to train the dataset. And lastly, Accuracy & Precision is self-explanatory. Based on all of these comparisons, a convolutional neural network was chosen as the backbone of our image recognition design.

**Encryption Design Trades**

To ensure that our communications remain confidential, an encryption algorithm is needed. Encryption is a way of hiding information in a jumbled display so that the contents are unreadable without the decryption key. Put simply, if someone does not have the key, they cannot interpret the contents. As the cybersecurity field has progressed, people have found ways to break encryption algorithms. Algorithms with smaller keys have been subject to exploitation. To choose the ideal algorithm, the group compared multiple encryption algorithms with common criteria to determine which algorithm is the most practical for implementation.

**Encryption Design Trades Table**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Option**  **[Req]** | **Key Length**  **[80]** | **Encryption Speed**  **[90]** | **Secure**  **Low, Medium, High**  **[Opt]** | **Tutorials [Opt]** | **Known Attacks**  **[Opt]** | **Time Required to Check all Possible Keys at 50 billion/Sec**  **[Opt]** | **Raspberry Pi Compatible**  **With OpenSSL**  **[Opt]** |
| AES | 128/192/256 [1] | Fast [1] | High [1] | [5], [6] | Side Channel Attack [2] | 5x10^21 days [1] | Yes [10], [11] |
| DES | 56 [1] | Slow [1] | Low [1] | [6] | Brute Force Attack [2] | 400 days [1] | Yes  [12] |
| 3DES | 112/ 168 [1] | Slow [1] | Low [1] | [7] | Multiple [2] | 800 days [1] | Yes  [13] |
| Blowfish | 448 [1] | Fast [1] | Medium [1] | [8] | Dictionary Attack [2] | 3200 days [1] | Yes [14] |
| IDEA | 128 [3] | Slow [3] | Medium [3] | [9] | Meet in the Middle Attack [4] | N/A | Yes [15] |

**Encryption Design Explanation:**

The encryption algorithms were compared based on key length, speed, security, tutorials available, known attacks, time required to brute force, and OpenSSL compatibility. The ratings indicate the capabilities and challenges of each algorithm, such as the speed which will be critical to our transmission. The chart provides an overview of the strengths and weaknesses of each encryption algorithm. Based on the previous comparisons, AES is the best option for this project due to its ideal key length, speed, security, tutorials, and compatibility.

# 2.c Proposed Solution

Based on the evaluated criteria above, the group decided to use Enhanced Shockburst for the communication, OpenCV for the death star image recognition, and the Advanced Encryption Standard (AES) for the encryption to ensure the confidentiality of the transmissions

# 2.d Impact of Success

**Cultural**

Positive: The potential positive impacts of this design include a potential shift in government and local governments may operate within a weaker regulatory environment.

Negative: The potential destruction of public works could normalize vandalism and subversive activities against legitimate authorities, which could negatively impact the culture overall for years to come.

**Economics**

Positive: The construction of public works may continue for longer leading due to the potential destruction of the space station, which may lead to higher employment rates within society and benefit resource extraction industries.

Negative: There is a massive potential for depreciation of the space station. Taxes will likely continue to be imposed upon populations that do not directly benefit from the construction of new public works, which may inhibit other economic activities. Insurance rates are likely to increase.

**Environmental**

Positive: Organisms with strong dorsal instincts are no longer impacted by gravity resulting from the mass of the space station; the artificial tides created by the space station give way to historic tidal norms. The night sky would be darker due to the reduction of light reflected off the surface of the space station.

Negative: Deorbiting debris from the space station may potentially contaminate sensitive ecosystems with heavy metals; dust introduced into the atmosphere from debris impacts may artificially prolong winters on nearby worlds. The sky will be visually less appealing due to the absence of the orbital space station—a severe reduction of ambience.

**Public health**

Positive: Public health is not improved by the destruction of infrastructure.

Negative: Those aboard the space station may be impacted by loss of life; survivors may be impacted by elevated rates of certain cancers. In the short term, there may be a strain on health care facilities.

**Public safety**

Positive: Certain populations within rebel-controlled territories may continue to have firm ground beneath their feet.

Negative: The debris from the space station may impact nearby space travel; the debris may be captured by nearby biospheres and fall upon civilian populations. The public aboard the space station may cease to exist.

**Public welfare**

Positive: New benefits may be provided to members of the imperial armed forces and their families to prevent desertion.

Negative: Assistance may need to be provided to individuals impacted by the debris resulting from the space station; the disruption in nearby supply chains may result in localized famine events.

**Social**

Positive: The destruction of the space station would result in a morale boost for rebel forces.

Negative: The loss of personnel aboard the space station may irreparably shift the culture and create resentment for alliance forces.

# 3. Context of Design Solution

# 3.a Design Objectives

1. The supplied USB that consists of the 100 PNG images will be able to decipher the 10 images of the death star.
   * Accurately distinguishing the images refers to the Raspberry Pi’s ability to examine all 100 images, disregard the 90 images that are not the death star, and then verify the death star images include weaknesses.
2. The data must be transmitted quickly from the Raspberry Pi to the rebel server.
   * Quick data transmission means our system will facilitate all necessary actions in no more than 600 seconds. The transmission process occurs from the Raspberry Pi to the Linux server. The Rebel server will be a Linux server. Failing to transmit the data in less than 600 seconds will result in detection of the system.
3. The data will be transmitted securely and then be properly verified.
   * Transmitting the data securely means our system will send data over an unmonitored communication protocol. Monitored communications include Wi-Fi, Bluetooth, cellular, and IO Ports. When the transmission process between the Raspberry Pi and the Linux server occurs, the data will be verified. Properly verifying the data means the MD5 Checksum will be sent back to the Raspberry Pi to verify the data arrived as expected.
4. The Linux Server will filter out the weaknesses of the 10 PNG death star images.
   * The 10 images of the death star have red circles indicating weaknesses. Filtering out the weaknesses includes using software to zoom in on the death star images and remove everything except the red circles on the death star. Doing so will specifically show where precisely the weakness is.
5. The Linux Server will broadcast the 10 weaknesses.
   * Broadcasting the received images means that once the Linux Server interprets the death star weaknesses, the weaknesses will be displayed on a mobile application for all to see. Doing so exposes the weaknesses and completes the mission.

# 3.b Design Assumptions

1. The Linux Server will have enough available USB-A ports to connect all necessary peripheral devices.
2. A functional server running a recent version of Linux with current drivers and sufficient available storage space for all necessary software and compilers will be provided.
3. A functional Raspberry Pi with up-to-date firmware and sufficient storage will be provided.
4. If the transmission process is completed in 10 minutes or less, the imperial guards will not become suspicious.
5. The room housing the server in the basement of the Wright State Russ Engineering Center will have a transparent glass window.
6. The 10 death star images will be visually distinct from non-death star images.
   * Visually distinct means that the 10 death star images will have a dark background with a solid image. No other images will have these properties.
7. Both the sending side of the glass and the server side will have access to standard American power outlets.
8. There will not be unusual interference or noise from other systems during the transferring of the images.
   * Unusual interference or noise refers to activity that is abnormal to the standard daily operations of the Russ Engineering Center.
9. The provided USB will contain only 100 1024x1024 8-bit-per-color-channel PNG images, 10 of which will contain known death star vulnerabilities demarcated by a red circle.

# 3.c Design Requirements

1. The design shall include a Raspberry Pi.
2. The design shall include a Linux Server.
3. The system shall include a communication device attached to the Raspberry Pi.
4. The system shall include a communication device attached to the Linux Server.
5. Software running on the Raspberry Pi shall read 100 PNG images stored on a USB containing 100 1024x1024 PNG images.
6. Software running on the Raspberry Pi shall discriminate between the 10 death star images containing red-circled weaknesses, and the 90 non-death star images.
7. The communication device attached to the Raspberry Pi shall transmit the 10 death star images to the Linux server through a wall containing a transparent window and a closed door.
8. All transmissions between the communication devices must be encrypted using an encryption algorithm with a key length greater than or equal to 64 bits.
9. The Raspberry Pi shall successfully transfer and verify the 10 death star images to the Linux server within the 600 second timeframe.
10. The integrity of the transmitted images shall be verified between the Raspberry Pi software and Linux Server via an MD5 checksum.
11. The system shall retransmit each image where the checksum does not match until all the images are successfully transferred and verified.
12. The Linux server shall crop the 10 death star images to include only the red-circled weakness for each image.
13. The weaknesses shall be exported to an Android or iOS mobile application and displayed in a horizontal grid.

# 3.d Design Constraints

1. There will be no physical link joining the Raspberry Pi to the Linux server.
2. The solution will not physically contact the boundary window.
3. The wireless channel between the Raspberry Pi and Linux server will not make use of Wi-Fi, Bluetooth, Cellular, and associated protocols.
4. The total bill of the materials will not exceed $300 USB (Excluding Standard)
5. All components of the system must be provided by or purchased through the course instructors.
6. The rebel server must be no closer than 5 meters to the glass wall.
7. The solution will not impede other network activities occurring concurrently with transception.
8. The established communication technique must be bidirectional.
9. All communication between the Pi and server must be encrypted.
10. The Raspberry Pi must transfer all 10 1024x1024 images of the Death Star within a total time of 600 seconds.

# 3.e Design Standards

1. Advanced Encryption Standard (AES)
2. IEEE C95.7-2022 – Standard for Electromagnetic Energy Safety Programs
3. IEEE 802.11-2024 – Standard for Information Technology Telecommunications and Information Exchange Between Systems Local and Metropolitan Area Networks – Specific Requirements – Part 11: Wireless Local Area Network (LAN) Medium Access Control (MAC) and Physical Layer (PHY) Specifications
4. ANSI C63.4-2014 – American National Standard for Methods of Measurement of Radio-Noise Emissions from Low-Voltage Electrical and Electronic Equipment in Range of 9 kHz to 40 GHz
5. IEEE 1722-2016 – Standard for a Transport Protocol for Time-Sensitive Applications in Bridged Local Area Networks
6. IEEE C2-1990 – National Electric Safety Code
7. ISO/IEC 15948:2004 – Information Technology – Computer Graphics and Image Processing – Portable Network Graphics (PNG): Functional Specification 80. IETF RFC 9110 – HTTP Semantics

# 3.f Design Functionality

1. Image Recognition

The system identifies which images are the relevant Death Star weaknesses and isolates those images for transmission. These images are also later cropped down to exclusively the highlighted weakness

2. Encryption

The system utilizes an AES-128 encryption algorithm which is performed on all data transmitted between the Linux server and the Raspberry Pi.

3. Transmission

The system transmits images and checksums bidirectionally between the Raspberry Pi and the Linux server over the wireless channel using the nRF52840 SOCs. This communication occurs exclusively over ESB; it does not communicate via Wi-Fi, Bluetooth, or cellular.

4. Mobile Application

Once transmission and image cropping has been completed, the designed mobile application connects to the server and displays the final weaknesses in table format. This connection is done over Wi-Fi.

# 3.g User Characteristics

The intended user should be able to lift at least five kilograms and have enough dexterity to plug in the USB stick and supply power to the system. The intended user of the system will possess a basic understanding of Linux systems with regards to navigation and executing programs. The intended user will also possess a basic understanding of installing mobile applications and navigating their interfaces.

# 3.h Operating Environment

The operating environment should have ready access to mains electricity with standard North American outlets. The operating environment should be free from spurious electromagnetic interference. The operating environment should be dry, sere, and the temperature should be within the range of -40 to 85℃.

# 4. Technical Approach

# 4.a Hardware/Software Tradeoffs

**Wireless Transmission Hardware Tradeoffs**

With the previously outlined decision to implement Enhanced ShockBurst protocol (ESB), the following table outlines various hardware options for implementation of this protocol. These physical transceivers have various differing characteristics as far as connectivity and hardware specifications that impact the scale of performance. The most commonly used connections are either USB, which can be simply plugged into an available USB port, or serial peripheral interface (SPI) which requires direct wiring to pins of the connected device.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Option**  **[Req]** | **Rated Data Throughput [90]** | **Rated Transmission Distance**  **[70]** | **Cost** | **Native Encryption Support**  **[80]** | **Hardware Interface**  **[30,40]** |
| NRF24L01 | 250Kbps / 1Mbps / 2Mbps [2] | 100 meters [2] | $1.39 [1] | None [2] | SPI [A] |
| Adafruit Feather nRF52840 Express | 250Kbps / 1Mbps / 2Mbps [4] | 100 meters  [4] | $24.95[4] | None [4] | SPI, UART, I2C, USB peripheral [4] |
| NRF52840-DONGLE | 1Mbps / 2Mbps [5] | Unknown | $10.00 [6] | AES-128 Accelerator [5] | USB [7] |
| SparkFun Pro nRF52840 Mini | 1Mbps / 2Mbps [8] | 300 meters / 100 meters [8] | $32.50 [9] | AES-128 [8] | SPI, UART, I2C, USB peripheral [8] |
| Nordic Thingy:91 | 250Kbps / 1Mbps / 2Mbps [10] | Unknown | $126.25 [11] | AES-128 [10] | SPI, UART, I2C, USB peripheral [10] |

The NRF52840-DONGLE is preferred due to its low cost and USB interface, which lends itself to interface both with the Raspberry Pi and the Rebel server. The Adafruit Feather Express and SparkFun Pro mini are of secondary preference due to their intermediate cost; they do have the benefit of providing a rated transmission distance. The Nordic Thingy:91 is not preferred due to its high unit cost. The NRF24L01 has a favorable unit cost, but it may potentially require more care to implement due to its lack of a USB interface.

**Image Recognition Software Tradeoffs**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Option**  **[Req]** | **Memory Requirements**  **[60,120]** | **Speed & Efficiency [90]** | **Ease of Integration**  **[Opt]** | **Compatibility**  **[60,120]** | **Cost ($)**  **[Opt]** |
| OpenCV | 256 MB RAM minimum [14] | Average [1] | Easy [1] | Cross-Platform  [2] | Free [7] |
| TensorFlow | 4 GB RAM [3] | Average [3] | Average [3] | Cross-Platform  [8] | Free [8] |
| YOLO | 388 MB RAM [15] | Fast [4] | Easy [4] | Cross-Platform  [9] | Free [9] |
| Amazon Rekognition | Cloud Based, Minimal System Requirements [5] | Average [5] | Easy [5] | Cloud-Based  [5] | ~ $0.0010 per image [6] |
| PyTorch | 4 – 8 GB RAM [14] | Slow [10] | Average [11] | Cross-Platform [11] | Free [11] |

This table compares several different image recognition tools based on several factors such as memory requirements, speed, ease of integration, compatibility, and cost. Tools like OpenCV and YOLO are considered “fast” because they can process images in real-time or near real-time. This is critical for tasks like quick object detection using as few resources as possible. TensorFlow and PyTorch are classified as “average” due to the fact they are more complex but may take longer to compute the data involved and process the image. Ease of integration defines how documented and how many API’s and prebuilt functions (such as programming languages or data libraries) are available. Tools like OpenCV and Amazon Rekognition are well documented API’s, prebuilt functions, and support. Tools like YOLO and PyTorch may need more knowledge and (or) setup. These classifications are based off how efficiently the tools handle these tasks and how quickly they can be set up and implemented with as few issues as possible. Seeing that YOLO wins in every category against the other options, we have decided to go with YOLO. It is easy to use, fast, and accurate.

**Encryption Software Tradeoffs**

Since the group decided to use AES encryption, a library to run the algorithm on is needed. OpenSSL is an open-source command line tool. LibgCrypt is a general-purpose crypto library. PyCrypto is a python cryptography toolkit used for hashing and encryption. GnuPG is an open-source implementation of OpenPGP. Put simply, all these options are libraries that could be used to implement AES. The group wants to ensure compatibility and optimal implementation for the design.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Option**  **[Req]** | **Compatible on Raspberry Pi [80]** | **Tutorials Available [Opt]** | **License**  **[Opt]** | **Overhead**  **[Opt]** |
| OpenSSL | Yes [1] | [2], [3], [11] | Apache 2.0 [7] | 10 bits [Team Experiment] |
| LibgCrypt | Yes [4] | [4], [12] | LGPLv2.1+ [8] | 26 bits [Team Experiment] |
| PyCrypto | Yes [5] | [5], [13] | Partially Public Domain, Partially Under BSD 2-Clause license [9] | 26 bits [Team Experiment] |
| GnuPG | Yes [6] | [6], [14] | GPL [10] | 71 bits [Team Experiment] |
| NRF52840-DONGLE | Yes [16] | [15] | N/A | N/A |

The encryption libraries are compared based on compatibility with the Raspberry Pi, tutorials available, licensing, and the change in file overhead. These evaluations help compare the capabilities and challenges of each method, such as the file overhead changes which will directly impact the speed of the transmission. The team wanted to conduct testing to evaluate the overhead. The team experiment is seen below:

**4.b Encryption** **Library Test**

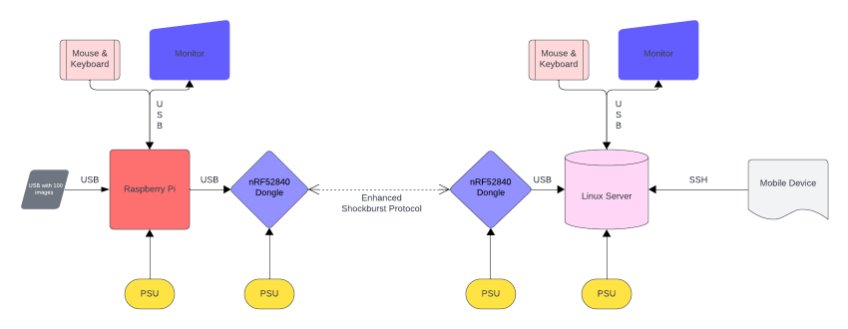
A screenshot of a computer

Description automatically generated

For the experiment,each of the encryption libraries encrypted the same file, with AES-128. Each file has the same basic contents inside.The hashes at the bottom prove that each algorithm encrypted the same file.After encrypting, the difference in the new file’s overhead from the original file is compared in the overhead column above. Based upon the collected data, it was determined that OpenSSL was the best option for the encryption library.

# 4.c Hardware Design

**Hardware Diagram**

**Hardware functional block definitions:**

1. USB with 100 images – Contains 100 1024x1024 pixel PNG images. 10 images depict the Death Star out of the 100 images on the USB Thumb drive.
2. Raspberry Pi - Software running on the Raspberry Pi must identify which 10 images depict the Death Star out of the 100 images on the USB Thumb drive.
3. PSU - Power supply unit. Supplies power to the individual components
4. Transceiver - A device that can both transmit and receive communications, in particular a combined radio transmitter and receiver. Will be used to transmit images to the Linux server.
5. Monitor – Displays visual information to the user.
6. Mouse & Keyboard – Allows for user input to the connected system
7. Linux Server – Isolates the weaknesses of the 10 images of the death star and uploads them to a web server to be displayed on the mobile application.
8. Mobile Device – Displays the weaknesses uploaded to the website from the Linux server

**4.d Software** **Design**

**Software Diagram**

A diagram of a computer hardware system

Description automatically generated**Software functional block definitions:**

1. Image Recognition Software – Software will analyze and filter the 10 death star images out of the 100 provided
2. Image Raw Data Conversion – The 10 death star images then have their raw data extracted to be sent
3. Data Encryption – The data of those 10 images is then encrypted to ensure secure transfer
4. Transceiver – Software communicates with the transceiver to send/receive data
   1. 4 → 5 – Encrypted image data is sent from the Raspberry Pi to the Linux Server
5. Transceiver – Software communicates with the transceiver to send/receive data
   1. 5 → 4 – MD5 Checksum of received images is sent from Linux Server to Raspberry Pi
6. Data Decryption – The received data is decrypted by the Linux server
7. Raw Data to Image – The raw data is converted back to 10 png images
   1. 7 – 5 – MD5 Checksum of received images is calculated and sent back to the transceiver
8. MD5 Checksum Validation – The 10 received images have their MD5 checksums checked against the original 10 images
9. Image Cropping Software – The 10 images are analyzed and cropped down to the circled weaknesses.
10. Server Hosts Images – The 10 cropped weakness images are hosted on the Linux server for access by the mobile app.
11. Mobile App – A mobile application for Android connects to the Linux server over Wi-Fi and displays the 10-weakness images.

**4.e Risks and** **Mitigations**

**Risk 1: Parts not arriving on time**

**Mitigations: Order the parts ahead of time**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Unmitigated:** | | | **Mitigated:** | | |
|  | Severity |  |  | Severity |  |
| Likelihood | |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  | **1** | **2** | **3** | **4** | **5** | | **1** |  |  |  |  |  | | **2** |  |  |  |  |  | | **3** |  |  | **x** |  |  | | **4** |  |  |  |  |  | | **5** |  |  |  |  |  | | | Likelihood | |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  | **1** | **2** | **3** | **4** | **5** | | **1** |  |  |  |  |  | | **2** |  | **x** |  |  |  | | **3** |  |  |  |  |  | | **4** |  |  |  |  |  | | **5** |  |  |  |  |  | | |

**Risk 2: Risk of fire.**

**Mitigation: Take proper safety measures (i.e.: gloves, fire extinguishers, etc.)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Unmitigated:** | | | **Mitigated:** | | |
|  | Severity |  |  | Severity |  |
| Likelihood | |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  | **1** | **2** | **3** | **4** | **5** | | **1** |  |  |  |  | **x** | | **2** |  |  |  |  |  | | **3** |  |  |  |  |  | | **4** |  |  |  |  |  | | **5** |  |  |  |  |  | | | Likelihood | |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  | **1** | **2** | **3** | **4** | **5** | | **1** |  | **x** |  |  |  | | **2** |  |  |  |  |  | | **3** |  |  |  |  |  | | **4** |  |  |  |  |  | | **5** |  |  |  |  |  | | |

**Risk 3: Someone gets sick or has an emergency for an extended time.**

**Mitigation: Communicate via discord, work from home wherever possible, stay up with hygiene, etc.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Unmitigated:** | | | **Mitigated:** | | |
|  | Severity |  |  | Severity |  |
| Likelihood | |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  | **1** | **2** | **3** | **4** | **5** | | **1** |  |  |  |  |  | | **2** |  |  |  |  |  | | **3** |  |  |  | **x** |  | | **4** |  |  |  |  |  | | **5** |  |  |  |  |  | | | Likelihood | |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  | **1** | **2** | **3** | **4** | **5** | | **1** |  |  |  |  |  | | **2** |  |  |  |  |  | | **3** | **x** |  |  |  |  | | **4** |  |  |  |  |  | | **5** |  |  |  |  |  | | |

**Risk 4: Raspberry PI is not fast enough to distinguish the 10 Death Star Images on test day**

**Mitigation: Have a backup module on standby if the speeds on our primary RF module are not fast enough**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Unmitigated:** | | | **Mitigated:** | | |
|  | Severity |  |  | Severity |  |
| Likelihood | |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  | **1** | **2** | **3** | **4** | **5** | | **1** |  |  |  |  |  | | **2** |  |  |  |  |  | | **3** |  |  |  |  |  | | **4** |  |  |  |  |  | | **5** |  |  |  |  | **x** | | | Likelihood | |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  | **1** | **2** | **3** | **4** | **5** | | **1** |  |  |  |  |  | | **2** |  |  | **x** |  |  | | **3** |  |  |  |  |  | | **4** |  |  |  |  |  | | **5** |  |  |  |  |  | | |

**Risk 5: The Raspberry PI does not work with the Dongle NRF52840 communication device**

**Mitigation: Our group is planning to order backup communication devices in the case that this happens**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Unmitigated:** | | | **Mitigated:** | | |
|  | Severity |  |  | Severity |  |
| Likelihood | |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  | **1** | **2** | **3** | **4** | **5** | | **1** |  |  |  |  |  | | **2** |  |  | **x** |  |  | | **3** |  |  |  |  |  | | **4** |  |  |  |  |  | | **5** |  |  |  |  |  | | | Likelihood | |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  | **1** | **2** | **3** | **4** | **5** | | **1** |  |  |  |  |  | | **2** | **x** |  |  |  |  | | **3** |  |  |  |  |  | | **4** |  |  |  |  |  | | **5** |  |  |  |  |  | | |

**Risk 6: There is too much interference and data transmission has errors**

**Mitigation: Simulate the same environment to see how much interference there is and plan accordingly**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Unmitigated:** | | | **Mitigated:** | | |
|  | Severity |  |  | Severity |  |
| Likelihood | |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  | **1** | **2** | **3** | **4** | **5** | | **1** |  |  |  |  |  | | **2** |  |  |  |  |  | | **3** |  |  |  |  | **x** | | **4** |  |  |  |  |  | | **5** |  |  |  |  |  | | | Likelihood | |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  | **1** | **2** | **3** | **4** | **5** | | **1** |  |  |  |  |  | | **2** | **x** |  |  |  |  | | **3** |  |  |  |  |  | | **4** |  |  |  |  |  | | **5** |  |  |  |  |  | | |

**Risk 7: Communication is not established**

**Mitigation: Check code and dongle in advance to see what the issue is. Use backup dongle if need be.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Unmitigated:** | | | **Mitigated:** | | |
|  | Severity |  |  | Severity |  |
| Likelihood | |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  | **1** | **2** | **3** | **4** | **5** | | **1** |  |  |  |  |  | | **2** |  |  |  |  |  | | **3** |  |  |  |  | **x** | | **4** |  |  |  |  |  | | **5** |  |  |  |  |  | | | Likelihood | |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  | **1** | **2** | **3** | **4** | **5** | | **1** |  |  |  |  | **x** | | **2** |  |  |  |  |  | | **3** |  |  |  |  |  | | **4** |  |  |  |  |  | | **5** |  |  |  |  |  | | |

**Risk 8: Images are not pulled from the USB**

**Mitigation: Check USB to ensure it’s plugged in, and check code in advance to troubleshoot issue.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Unmitigated:** | | | **Mitigated:** | | |
|  | Severity |  |  | Severity |  |
| Likelihood | |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  | **1** | **2** | **3** | **4** | **5** | | **1** |  |  |  |  |  | | **2** |  |  |  |  |  | | **3** |  |  |  |  | **x** | | **4** |  |  |  |  |  | | **5** |  |  |  |  |  | | | Likelihood | |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  | **1** | **2** | **3** | **4** | **5** | | **1** |  |  |  |  | **x** | | **2** |  |  |  |  |  | | **3** |  |  |  |  |  | | **4** |  |  |  |  |  | | **5** |  |  |  |  |  | | |

**Risk 9: Images are not filtered from the USB**

**Mitigation: Simulate in advance to address any issues beforehand.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Unmitigated:** | | | **Mitigated:** | | |
|  | Severity |  |  | Severity |  |
| Likelihood | |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  | **1** | **2** | **3** | **4** | **5** | | **1** |  |  |  |  |  | | **2** |  |  |  |  |  | | **3** |  |  |  |  | **x** | | **4** |  |  |  |  |  | | **5** |  |  |  |  |  | | | Likelihood | |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  | **1** | **2** | **3** | **4** | **5** | | **1** |  |  |  |  | **x** | | **2** |  |  |  |  |  | | **3** |  |  |  |  |  | | **4** |  |  |  |  |  | | **5** |  |  |  |  |  | | |

**Risk 10: Image encryption for transmission breaks the transmission.**

**Mitigation: Simulate in advance, have another protocol on standby and ready to use in the event our primary protocol doesn’t work.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Unmitigated:** | | | **Mitigated:** | | |
|  | Severity |  |  | Severity |  |
| Likelihood | |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  | **1** | **2** | **3** | **4** | **5** | | **1** |  |  |  |  |  | | **2** |  |  |  |  |  | | **3** |  |  |  |  | **x** | | **4** |  |  |  |  |  | | **5** |  |  |  |  |  | | | Likelihood | |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  | **1** | **2** | **3** | **4** | **5** | | **1** |  |  |  |  | **x** | | **2** |  |  |  |  |  | | **3** |  |  |  |  |  | | **4** |  |  |  |  |  | | **5** |  |  |  |  |  | | |

**Risk 11: Hash cannot be verified**

**Mitigation: Resend the hash**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Unmitigated:** | | | **Mitigated:** | | |
|  | Severity |  |  | Severity |  |
| Likelihood | |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  | **1** | **2** | **3** | **4** | **5** | | **1** |  |  |  |  |  | | **2** |  |  |  |  |  | | **3** |  |  | **x** |  |  | | **4** |  |  |  |  |  | | **5** |  |  |  |  |  | | | Likelihood | |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  | **1** | **2** | **3** | **4** | **5** | | **1** |  |  | **x** |  |  | | **2** |  |  |  |  |  | | **3** |  |  |  |  |  | | **4** |  |  |  |  |  | | **5** |  |  |  |  |  | | |

**Risk 12: Incorrect images are transmitted and displayed which takes up time.**

**Mitigation: Test this scenario in advance and note any issues. Address any issues before the final demo.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Unmitigated:** | | | **Mitigated:** | | |
|  | Severity |  |  | Severity |  |
| Likelihood | |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  | **1** | **2** | **3** | **4** | **5** | | **1** |  |  |  |  |  | | **2** |  |  |  |  |  | | **3** |  |  | **x** |  |  | | **4** |  |  |  |  |  | | **5** |  |  |  |  |  | | | Likelihood | |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  | **1** | **2** | **3** | **4** | **5** | | **1** |  |  |  |  |  | | **2** |  | **x** |  |  |  | | **3** |  |  |  |  |  | | **4** |  |  |  |  |  | | **5** |  |  |  |  |  | | |

**Risk 13: OpenCV does not communicate with the server and filter the images**

**Mitigation: Test in advance and have a backup option that we have researched on standby**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Unmitigated:** | | | **Mitigated:** | | |
|  | Severity |  |  | Severity |  |
| Likelihood | |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  | **1** | **2** | **3** | **4** | **5** | | **1** |  |  |  |  |  | | **2** |  |  |  |  |  | | **3** |  | **x** |  |  |  | | **4** |  |  |  |  |  | | **5** |  |  |  |  |  | | | Likelihood | |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  | **1** | **2** | **3** | **4** | **5** | | **1** |  |  |  |  |  | | **2** | **x** |  |  |  |  | | **3** |  |  |  |  |  | | **4** |  |  |  |  |  | | **5** |  |  |  |  |  | | |

**Risk 14: Mobile app doesn’t connect to the server to receive images.**

**Mitigation: Perform advanced testing to ensure the browser properly handles the 10 images**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Unmitigated:** | | | **Mitigated:** | | |
|  | Severity |  |  | Severity |  |
| Likelihood | |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  | **1** | **2** | **3** | **4** | **5** | | **1** |  |  |  |  |  | | **2** |  |  |  |  |  | | **3** |  |  |  |  |  | | **4** |  |  | **x** |  |  | | **5** |  |  |  |  |  | | | Likelihood | |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  | **1** | **2** | **3** | **4** | **5** | | **1** |  |  |  |  |  | | **2** |  | **x** |  |  |  | | **3** |  |  |  |  |  | | **4** |  |  |  |  |  | | **5** |  |  |  |  |  | | |

# 5. Appendix:

5.a Gant ChartA screenshot of a computer

Description automatically generated

# 5.b Budget

A close-up of a list

Description automatically generated

# 5.c Requirements Testing

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Requirement | Test Method | Evaluation Method | Threshold | Objective |
| 10 | Inspection | Verify the presence of a Raspberry Pi within the solution. | The Raspberry Pi is present. | The Raspberry Pi is present. |
| 20 | Inspection | Verify the presence of the Linux Server within the delivered solution. | The Linux server is present. | The Linux server is present. |
| 30 | Inspection | Verify the presence of the communication device attached to the Raspberry Pi within the delivered solution. | The communication device is present | The communication device is present. |
| 40 | Inspection | Verify the presence of a communication device attached to the Rebel Server within the delivered solution. | The communication device is present. | The communication device is present. |
| 50.1 | Inspection | Verify that the device can detect USB connections from mass storage devices. | The USB connection is recognized. | Verify that software can recognize and load a 1024x1024 png from the USB device. |
| 50.2 | Test | Test that the software can recognize and load a 1024x1024 png from the USB device. | At least one image is recognized. | All images are recognized by the software. |
| 60 | Functional Test | Test that the software can differentiate between an image of the death star and non-death star images. | Software can classify a single image as a photo of the death star or not. | Given 100 images, 10 of which being death star images, the software can accurately differentiate and identify the 10 death star images. |
| 70.1 | Functional Test | Unidirectional communication between a transmitter and a receiver. | Receiver can recognize a connection from a transmitting device over air. | Receiver can parse data packets from a transmitting device over air. |
| 70.2 | Functional Test | The nRF52840 is connected to the computer via USB. A script opens a serial link with the device. The computer sends a block of data to the nRF52840 dongle as a query; the nRF52840 sends its own data block in its response. | Data blocks are transmitted between devices. | Establish bidirectional transmission of data between transceiver and computer via a USB interface. |
| 70.3 | Functional Test | An initiating transceiver sends a query to the reacting transceiver. When the reacting transceiver receives the query, it sends a response. The initiating transceiver awaits the response. | A response to the query is received by the initiating transceiver. | Establish bidirectional transmission of data between transceivers. |
| 70.4 | Functional Test | An initiating transceiver hashes a data block. The data block is then transmitted over ESB to the reacting transceiver. Upon reception of the data block, the reacting transceiver hashes the data block. A hash exchange then occurs. If the hashes disagree, retransmission occurs until the hashes agree. | The data block and hashes are in accord between transceivers. | Establish bidirectional transmission of data between transceivers with hash exchange. |
| 70.5 | Functional Test | An initiating transceiver possesses a series of data blocks. It hashes the data blocks. The initiating transceiver begins a process of transmitting a data block, the reacting transceiver then receives that data block, hashes it, and then a hash exchange occurs for that block; retransmission for a data block occurs until the hashes agree between transceivers. The process repeats until all data blocks have been received and their hashes are correct. | All data blocks and hashes are in accord between transceivers. | Establishes bidirectional transmission of sequentially packaged data with hash exchanges between transceivers. |
| 70.6 | Functional Test | An image existing on a computer is converted to a bitstream and sent to the initiating transceiver via USB. The initiating transceiver then hashes that data and transmits it to the reacting transceiver. The reacting transceiver hashes the data it received. A hash exchange occurs. Retransmission occurs until both the initiating and reacting transceivers agree that the hashes are in accord. Once this is true, the received data is reformatted as an image. | The image on the reacting transceiver end is identical to the image transmitted by the initiating transceiver and their hashes agree. | Establish communication of image data wirelessly with verification. |
| 70.7 | Integration Test | The initiating and reacting receivers are positioned on either side of the air gap. 10 images are communicated from the Raspberry Pi to the initiating transceiver as bitstreams via USB. The initiating transceiver hashes each image bit stream and then transmits the bit stream to the reacting transceiver. A hash exchange occurs. Retransmission occurs if the hashes disagree between transceivers. Once the hashes agree, the next image bit stream is transmitted. This continues until all image bits streams have been transmitted and verified via hashes. After an image bitstream has been verified, it is sent to the rebel server via USB and reformatted as an image. | All 10 received image files are identical to the transmitted ones and their hashes agree. | Establish viability of proposed communication system in its indented operating environment. The images can be communicated and verified over the channel and reconstructed on the reacting transceiver side. |
| 80.1 | Functional Test | Encrypt 1024x1024 PNG of death star on the Raspberry Pi utilizing a key length >= 64 bits. | Image is properly encrypted. | Image is properly encrypted. |
| 80.2 | Functional Test | Decrypt 1024x1024 encrypted PNG of death star on the Linux server utilizing a key length >= 64 bits. | Image is properly decrypted. | Image is properly decrypted. |
| 80.3 | Functional Test | Encrypt MD5 checksum on the Linux server utilizing a key length >= 64 bits. | Checksum is properly encrypted. | Checksum is properly encrypted. |
| 80.4 | Functional Test | Decrypt an encrypted MD5 checksum on the Raspberry Pi utilizing a key length >= 64 bits. | Checksum is properly decrypted. | Checksum is properly decrypted. |
| 90.1 | Functional Test | Transmit encrypted 1024x1024 PNG of the death star from the Raspberry Pi to the Linux Server. | Encrypted image is accurately received. | Encrypted image is accurately received. |
| 90.2 | Speed Test | Transmit 10 encrypted 1024x1024 PNG images of the death star from the Raspberry Pi to the Linux Server and measure time. | All 10 images are accurately transmitted in < 600 seconds. | All 10 images are accurately transmitted in < 300 seconds |
| 100.1 | Functional Test | Transmit an encrypted MD5 checksum of a received image from the Linux server to the /Raspberry Pi. | Raspberry Pi accurately receives the encrypted checksum. | Raspberry Pi accurately receives the encrypted checksum. |
| 100.2 | Functional Test | Compare the decrypted received checksum against the checksum of the original image transmitted. | Checksum matches expected value. | Checksum matches expected value. |
| 110 | Functional Test | Send an incorrect checksum from the Linux server to the Raspberry Pi and verify that the Pi resends the image. | The Raspberry Pi re-transmits the appropriate image. | The Raspberry Pi re-transmits the appropriate image. |
| 120 | Functional Test | Give the Linux server a 1024x1024 PNG image of the death star with a highlighted weakness and confirm that the software accurately crops the image down to the red circle. | Image is properly cropped to expected output. | Image is properly cropped to expected output. |
| 130.1 | Functional Test | The 10 cropped images are uploaded from the Linux server to the webserver of the mobile app. | The Linux server connects to the web server of the mobile app and uploads the images. | The 10 cropped images are successfully uploaded from the Linux server to the web server for the mobile app. |
| 130.2 | Functional Test | The mobile app accesses the web server and displays the grid of cropped images. | The web server displays the received images in a grid table. | The mobile app successfully accesses the web server and displays the grid of cropped images. |

**Constraints Testing**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Constraint | Test Method | Evaluation Method | Threshold | Objective |
| 10 | Visual Inspection | Visually inspect the Linux server and the Raspberry Pi to check for physical connection | No physical connection to the Linux Server | No physical connection to the Linux Server |
| 20 | Visual Inspection | Visually inspect the boundary windows to ensure nothing is connected to it. | No physical connection | No physical connection |
| 30 | Functional Test | Check the RF module to ensure it’s being used, and also check each part of the project to ensure that Bluetooth, cellular, nor any other protocols are being used. | Wi-Fi, Bluetooth, cellular, nor any other associated protocols is used | Wi-Fi, Bluetooth, cellular, nor any other associated protocols is used |
| 40 | Budget Inspection | Check our orders, and spreadsheet to ensure that we don’t exceed $300 USD. | Must not exceed budget of $300 USD | Verify that the total project shall not exceed $300 USD. |
| 50 | Rubric Inspection | Check our components and track where we got them from. | All materials must come from course instructors | All materials must come from course instructors |
| 60 | Visual Inspection | Measure the server to ensure that it’s at least 5 meters or longer from the window. | Is greater than or equal to 5 meters from glass | Is greater than or equal to 5 meters from glass |
| 70 | Functional Test | Use Wireshark or any other application to ensure that the project is not interfering with anything that was not intended to. | No disruption to other activities | No disruption to other activities, and no one knows about the data transmission. |
| 80 | Functional Test | Visually check that communication is being sent and received from the Raspberry Pi and the server. | Is bi-directional | Is bi-directional |
| 90 | Functional Test | Ensure that AES Encryption is being enforced during the transmission. | Data is encrypted | Data is encrypted |
| 100 | Functional Test | Time the total length of time that it takes the images to transfer to the server and verify it. | The 10 1024x1024 images of the death star within 600 seconds | The 10 1024x1024 images of the death star within 300 seconds |

**Standards Testing**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Standard | Test Method | Evaluation Method | Threshold | Objective |
| 10 | Analysis | Data is encrypted | Encryption performed accurately follows the outlined AES standards | All encrypted communication between systems adheres to AES standards of encryption |