Q&A Interface design for symmetric block cipher encryption primitives through handmade and ML decision trees

By Rohin Dasari, Brandon Vo, Alvin Crighton

Project Choice:

- For our project, we opted to create a decision tree and web interface.
- The cryptographic primitive we chose were symmetric encryptions of block ciphers. The following block cipher schemes were considered:
  - 3DES
  - AES-128
  - IDEA
  - SIMON
  - Threefish-1024
- We did research based on the design intent and results calculated from research papers regarding the block ciphers and read the design documents from the original designers of the cryptographic scheme.
- To further back our data, we acquired and implemented working libraries of each encryption scheme and ran benchmark tests to further validate our data regarding the block cipher schemes.

Notable features from each scheme: 3DES

- While it is still considered (mostly) secure and hasn't been broken entirely yet, 3DES will be officially deprecated after 2023 under NIST guidelines.
- This is in response to the Sweet32 attack which has discovered an exploit using the short block lengths of 3DES to discover a block collision.
- Regardless, it is still available and usable with legacy software and is compatible with applications that were not forcibly disabled by developers.
- Declared that 3DES was designed to prioritize security over speed due to the intent even if 3DES is noticeably weaker and slower than AES.

Noticeable Features: AES-128

- AES is widely considered to be the standard choice for symmetric encryption
  - Simple Operations = fast and efficient
  - more lightweight than its older encryption methods like 3DES
- AES has proven to be secure after remaining unbroken despite over two decades of widespread adoption and implementation.
- AES achieves nonlinearity and high diffusion through its 10-round operation, requiring an attacker to run 2^{190} operations in the best case scenario to break an AES encryption.

Noticeable Features: IDEA

- Was designed specifically as a counterpart to AES in the 2000s (Arbitrarily speaking, we will consider IDEA to be a, relatively speaking, new algorithm because it was officially designed and patented sometime between 2000 and 2004.
- More accessible for support and adoption as the patent expired in 2012.
- While AES focused on using simple operations that permeate for several rounds, IDEA uses complex operations to achieve stronger security while saving memory.
- However, IDEA is more dependent on its key, block length, and # of rounds as opposed to AES.
    - An encryption that lasts 6 rounds or less (8.5 round is the standard) is breakable.
- IDEA suffers heavily from a weak set of encryption procedures, meaning that weak keys will lead to a weak ciphertext.
- Estimated # of operations to crack IDEA: $2^{126.1}$

Noticeable Features: Simon

- SIMON is a (relatively-speaking) recently developed in 2013 by the NSA.
- Has a key length of 128 bits and operates for 44 rounds.
- This allows for weak, low-end hardware access to an effective encryption scheme on devices are unable to even run AES.
- While Simon may be slower than AES on standard devices, Simon excels in performance on resource-constrained devices compared to AES.
- Estimated number of operations needed to break Simon: $2^{125.7}$

Noticeable Features: Threefish-1024

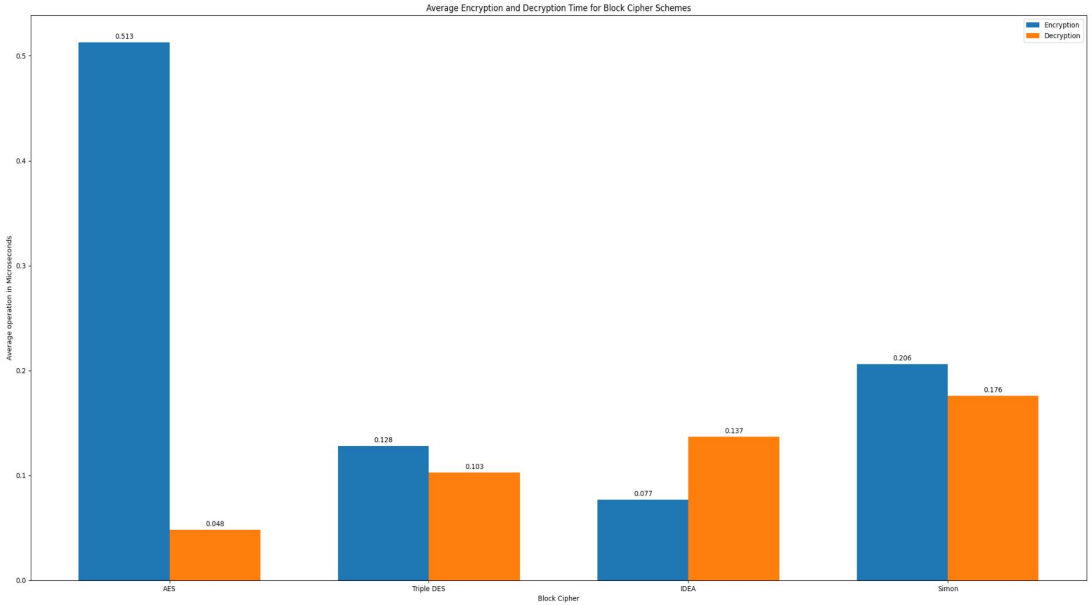
- Created in 2008 based off the Skein hash function where its predecessor, Twofish, was designed to compete against AES
- As opposed to AES and SIMON, Threefish was designed sacrifice as much memory and sometimes disk space to prioritize security.
- Keysize can go up to 1024 bits where its block encryption scheme will perform **80 rounds** of alternating XOR/addition procedures to achieve nonlinearity, allowing Threefish to become a pseudo-random permutation.
- Estimated # of operations to crack Threefish: $\mathbf{2^{222}}$. This is exponentially larger compared to the rest of the operations provided
- Despite being able to compete with AES, Threefish isn't widely available nor widely supported.
- Finding a functioning Threefish implementation was more challenging than finding a functioning library for the rest of the block cipher schemes.

Compiled Table

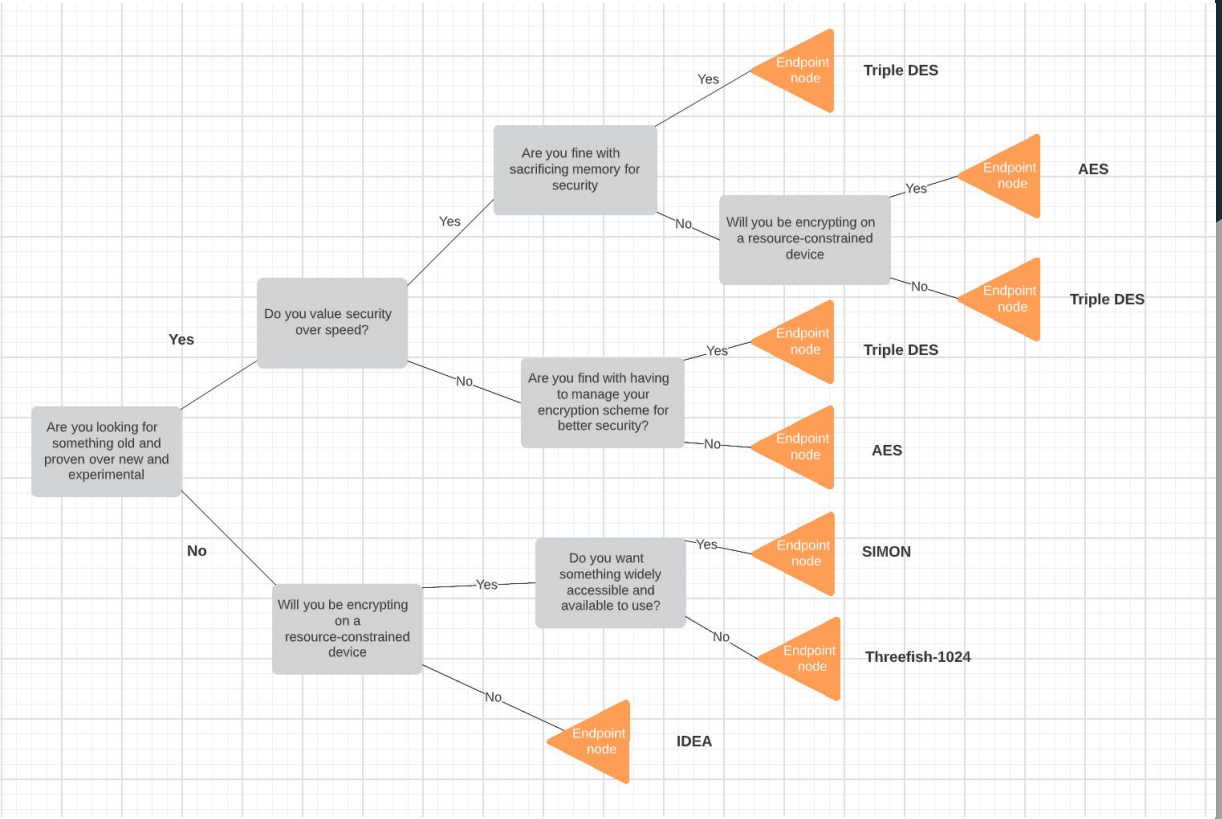
Number of operations to crack	Runtime (clock cycles per byte)	Block Size (Bits)	Old and proven or new and unknown?	Widely supported?	Complex or Simple	Encryption Scheme
2^{113}	108	64	Old (1975)	Until 2023	Simple	Triple DES
infeasible	18	18	Old (1998)	Yes	Simple	AES-128
$2^{125.7}$	7.5	Flexible	New (2013)	Yes	Simple	SIMON
$2^{126.1}$	132	64	New (~2000)	Yes	Complex	IDEA
2^{222}	881	1024	New (2008)	No	Complex	Threefish-1024

Runtime Analysis

- Some of the runtime values from the table above are informed by a runtime analysis we performed on some of the schemes
- Threefish-1024 is not shown since a stable, non-deprecated implementation could not be found or made



Decision Tree made by Hand



Decision Tree Implementation

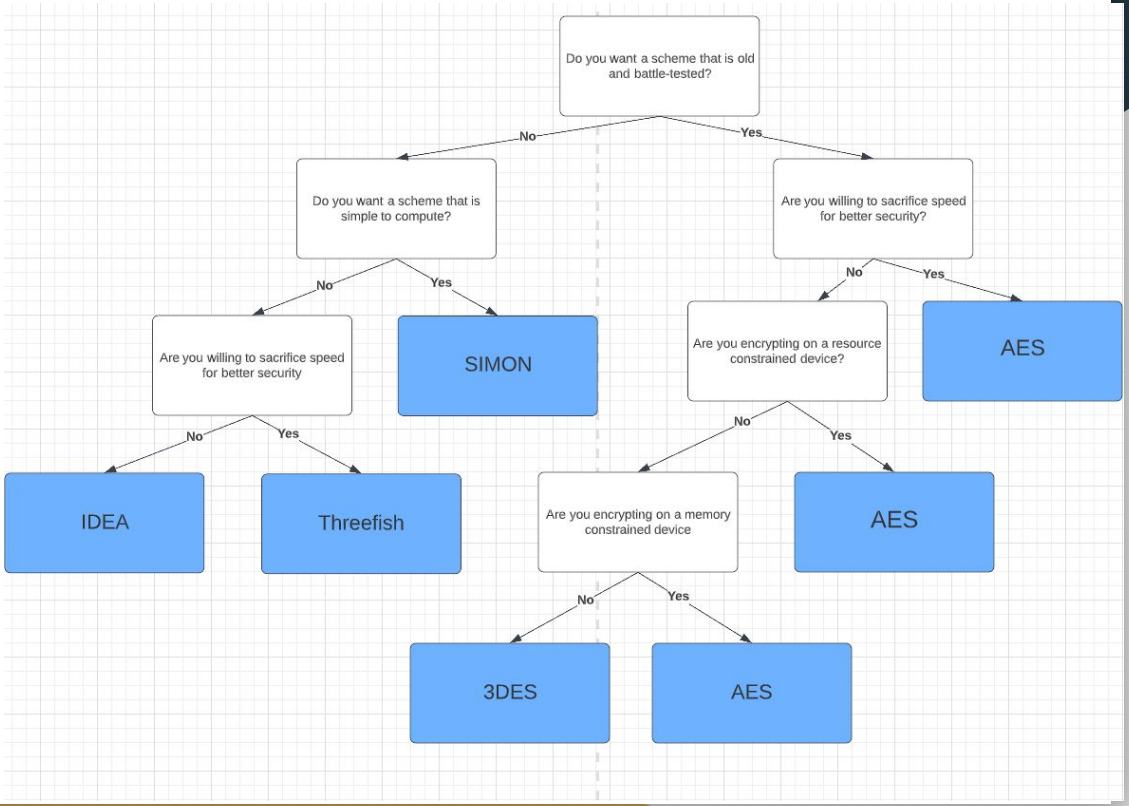
- Next, we created a set of decision trees by using a machine learning library from scikit.
- Using the table from before, we created a list of attributes that, when put together, would best describe one of the five block cipher schemes
- The table was slightly modified to produce a functional ML dataset
  - The table was encoded in such a way that each feature could be represented as binary values
  - Certain schemes can exhibit contradictory values in the table:
    - For example: AES can be used on both resource-constrained devices and non resource-constrained devices

(Abbreviated) Data Set (for an example, not actual use)

Old and proven over modern and experimental?	Do you want a widely supported, documented, and well known scheme?	Are you encrypting data on a resource-constrained device?	Security over speed?	Do you want a simple or complex scheme?	Are you fine with high memory usage?	Scheme output
Old	No	No	Security	Simple	Yes	Triple DES
Old	Yes	Yes	Speed	Simple	No	AES
Old	Yes	Yes	Security	Simple	No	AES
New	Yes	Yes	Security	Complex	No	IDEA
New	Yes	Yes	Security	Simple	No	SIMON
New	No	No	Security	Complex	Yes	Threefish

Decision Tree by Machine Learning

- This decision tree is 100% accurate based on the values used in the compiled table
- Less questions overall than handmade tree



Q&A Interface

- Using the information from the data and decision trees we implemented our Q&A interface by creating and hosting an online webpage on Github.io.
- The web page was built using HTML, Javascript through a library called ReactJS
- The decision tree itself was serialized and converted into a binary tree.
- The root node contains the first question and presents a binary choice for every succeeding question which directs the binary tree.
- Based on the user's input, the set of questions will be provided based on the direction of the binary tree's traversal.
- Once a leaf node has been reached, it will read the scheme found and provide the user's matched output.

Q&A Interface

Site: https://bvo4.github.io/CS6903ProjectSite/