Lab 3

Carl Benson

May 21, 2018

1 Main Idea

This algorithm seeks to create a way to securely and robustly log and report data while maintaining a minimal size footprint.

1.1 Compromise Resiliency

By mixing in previous values, steps 3-5 create a hash chain to show if a previously recorded value has been later modified.

1.2 All-or-Nothing Verification

Over time, step 5 creates a string of hashes, which will indicate if a previous data entry was modified.

1.3 Authentication and Integrity

Steps 2, 3, 4, 5, and 7 works towards achieving authentication and integrity. Steps 2-5 ensure the integrity of the data from attackers, while step 7 protects the integrity from loss during transmission.

1.4 Compression

The very first step of the algorithm is to run the data through a Huffman Compression. This compression step also provides minor benefits to other properties as well. Compressed data means a smaller size, which in turn requires less data to be transmitted and verified. This helps to reduce overhead in following steps by reducing the amount of data they need to process.

1.5 Aggregate Authentication

1.6 Packet Loss Resiliency

Compression of the data in the first step reduces the amount of packets that must be sent, which reduces the amount that can be lost. In the final step, the introduction of RID provides the primary source of resiliency to packet loss.

2 Order of Operations

Compression occurs first to give it the best chance of achieving a high efficiency compression result. Compression algorithms, particularly Huffman, rely on the statistical differences between the occurrences of characters. Proper encryption will return a result with uniform chance for a character to be used. This severely cripples the efficiency a compression algorithm can achieve.

Redundancy algorithms use certain properties in order to retain the contents of the message even after losing portions of it. Running a compression algorithm such as Huffman over these redundancies would weaken or remove these properties. If a portion of the Huffman encoding gets lost in transport, that portion is gone. Additionally, by compressing the data first, there is less data that needs to be protected from data loss.

Redundancy happens after encryption in order to best protect the encrypted data during transport. Similarly to compression, encrypting the redundancy negatively impacts the quality of redundancy that is possible.