A Huffman Code Based Crypto-System

Y. Gross, S. T. Klein, E. Opalinsky,
R. Revivo,D. Shapira

DCC 2022

Outline

Concerns of Communication over a network

1. processing speed
2. space savings of the transformed data
3. security

- **Data Compression**

  representation in fewer bits
- **Encryption**

  protecting information

  achieved by removing redundancies.

Combine for faster and safe transfer

COMRESSION CRYPTOSYSTEM

Compression Crypto-System

- Encrypt then compress - not possible
- Compress then Encrypt
- **Embed encryption into compression**

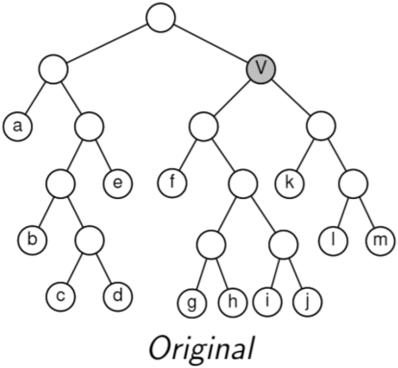
Cryptosystem based on Huffman Coding

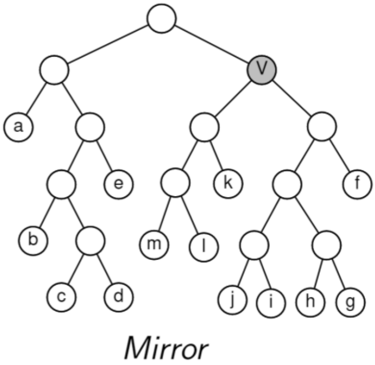
Initialize a Huffman tree according to the probabilities

- Use a secret key to select an internal node V
- Apply a *transformation*
- The transformation will *preserve* the codeword **lengths**
- Same compression ratio, different output for different keys

Transformation types: MIRROR

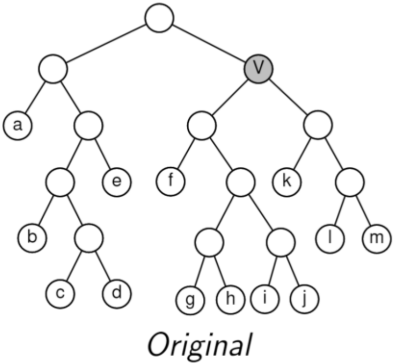
$$10110 \text{ (leaf i)} \Rightarrow \overline{10110} = 11001$$





Transformation types: SWAP

$$10110 \text{ (leaf i)} \Rightarrow 1\bar{0}110 = 11110$$



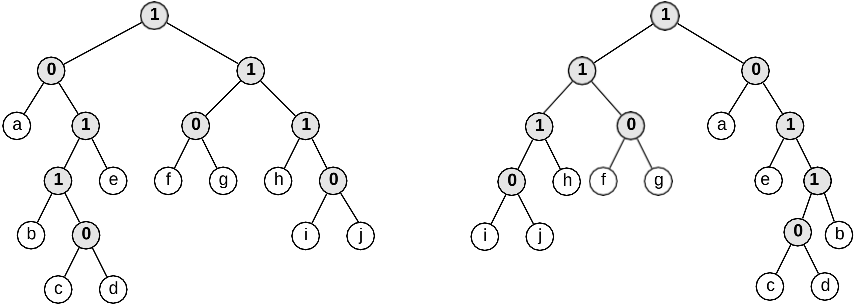
Transformations

- **additional** parameter : integer $1 \leq k < \sigma$
- choose k internal nodes $\{v_1, \ldots, v_k\}$ $\Rightarrow$ any transformation can be applied any k number of times (separately or combined)

  (use $k \log(\sigma)$ bits of the secret key)

- control the **trade-off between security and time complexity**

Transformations: BULK-CRYPTO-HUFFMAN



Example with the sub-bit stream 101101100 of a secret key.

Transformation types: Mirror and Swap

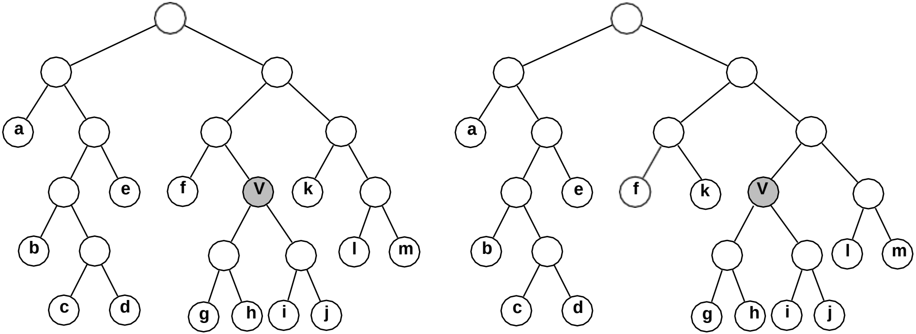
- easy to implement, but for the σ leaves of the tree,

  can only produce $2^{\sigma-1}$ permutations

  much less than **possible** $\sigma!$.

Transformation types: LEVEL-SWAP

1**01**10 (leaf i) => 1**10**10.



Security

Resistance to Cipher Attacks

Chosen Plaintext Attacks

The objective of encryption is to **hide** the content of a given plaintext file from an unauthorized eavesdropper.

The goal of such an opponent, on the other hand, is to try to **break the code**

Resistance to Cipher Attacks

Is the code breakable?

- Statistics of the occurrences of the alphabet symbols in natural languages are well-known.
- Guessing the length of codewords with high probability
- Decreases the number of partitions of the code into codewords?

Resistance to Cipher Attacks

Huffman Compression Crypto-System

- Apply transformations on Huffman tree constantly.
- Opponent knows about the details of the process except the secret key.
- This turns the problem of partitioning the ciphertext into codewords into difficult

Swap Mirror Prefix Code (SMPC) NP-Complete

Input: Given are positive integers ℓ, k and p, a text $T = x_1 x_2 \cdots x_n$ over some alphabet Σ of size $\sigma = |\Sigma|$, an initial Huffman tree H for Σ, a set of k transformations of the type swap or mirror, and a binary sequence S.

Question: Is there a subsequence S' of S of length $|S'| = \ell$, such that S' can be partitioned into codewords of the original Huffman code induced by H, so that

1. the lengths of the codewords belong to $\{s, \ldots, s+p\}$ for some integer s;
2. the set of these codewords satisfies the prefix property;
3. each codeword in S' consistently encodes a character of T;
4. one of the k transformations is applied to the current Huffman tree after the processing of some of the characters.

Chosen Plaintext Attacks

Same secret key, different ciphertexts.

- Add new symbol DC, as a *don't-care*, and $\Sigma^I = \Sigma \cup \{DC\}$
- Encoder adds DC in random locations.
- How many DC's not increasing the text significantly?

CPA security: DC symbol

- DC is added with probability $\frac{\log(i+1)}{c\,i}$, where $c > 1$ is a constant controlling the total number of inserted DCs.
- Expected distance between successive occurrences of DCs at position i by $\frac{c\,i}{\log(i+1)}$, as if we were using a constant probability between successive insertions of DCs.
- The overall expected distance between DCs for the entire range is then $E = \frac{1}{n}\sum_{i=1}^{n} \frac{c\,i}{\log(i+1)} = \theta(\frac{n}{\log n})$.
- Expected number of DCs, $\theta(\log n)$, is not bounded, and the *fraction* $\frac{\log n}{n}$ of inserted elements tends to zero.

CPA Security: skip

- Even with randomly inserted DCs, there could be a weakness in CPA security for Static Huffman.
  (same numbers of DC's)

- After a DC skip over a small constant number h of bits of the secret key.

Outline

The Algorithm

Algorithm 1: *Crypto-Huffman — Encoding*

CRYPTO-HUFFMAN-ENCODE($x_1 x_2 \cdots x_n,\ k,\ \mathcal{K},\ h$)

initialize the model

for $i \leftarrow 1$ **to** n **do**

  choose randomly a probability value p

  if $p < \frac{\log(i+1)}{c\, i}$ **then**

    encode DC according to the current model

    skip h bits in $\mathcal{K}$

  encode x_i according to the current Huffman tree $\mathcal{T}$

  use the secret key $\mathcal{K}$ to select k internal nodes $\{v_1, \ldots, v_k\}$ in $\mathcal{T}$

  for $j \leftarrow 1$ **to** k **do**

    apply *transformation* on v_j in $\mathcal{T}$

Algorithm 2: *Crypto-Huffman — Decoding*

CRYPTO-HUFFMAN-DECODE($y_1 y_2 \cdots y_m$, k, $\mathcal{K}$, h)

initialize the model
for $i \leftarrow 1$ **to** m **do**
    $x \leftarrow$ decoding of y_i according to the current model
    if $x =$ DC **then**
        skip h bits in $\mathcal{K}$
    else
        output x
        use the secret key $\mathcal{K}$ to select k internal nodes $\{v_1, \ldots, v_k\}$ in $\mathcal{T}$
        for $j \leftarrow 1$ **to** k **do**
            apply *transformation* on v_j in $\mathcal{T}$

Empirical Results

Data Set

Large Corpus taken from the Canterbury[1] corpora

bible.txt, the King James version of the Bible, of size 4,047,392 Bytes.

[1] http://corpus.canterbury.ac.nz

Empirical results: Uniformity

bit-str	STATIC	L-SWAP	MIRROR-1	MIRROR-2	SWAP-1	SWAP-2	BULK
0	0.495477	0.500097	0.500061	0.500091	0.500138	0.500092	0.500034
1	0.504523	0.499903	0.499939	0.499909	0.499862	0.499908	0.499966
00	0.249532	0.250076	0.249985	0.250030	0.250074	0.250030	0.250060
01	0.245944	0.250021	0.250076	0.250060	0.250074	0.250060	0.249974
10	0.245944	0.250021	0.250076	0.250060	0.250074	0.250060	0.249974
11	0.258579	0.249882	0.249863	0.249850	0.249798	0.249849	0.249992
000	0.122962	0.121652	0.124964	0.124989	0.125050	0.124989	0.125053
001	0.126570	0.128424	0.125022	0.125042	0.125024	0.125042	0.125007
010	0.119103	0.121522	0.125012	0.125093	0.124975	0.125093	0.124968
011	0.126842	0.128499	0.125063	0.124968	0.125089	0.124968	0.125007
100	0.126570	0.128424	0.125022	0.125042	0.125024	0.125042	0.125007
101	0.119374	0.121597	0.125054	0.125018	0.125040	0.125018	0.124967
110	0.126842	0.128499	0.125063	0.124968	0.125089	0.124968	0.125007
111	0.131737	0.121384	0.124800	0.124882	0.124709	0.124882	0.124985

Probability of 1-, 2- and 3-bit substrings for the Huffman variants.

Uniformity: KL distance

- Kullback–Leibler (KL) divergence
  $P = \{p_1, \ldots, p_n\}$, $Q = \{q_1, \ldots, q_n\}$,

$$D_{KL}(P \| Q) = \sum_{i=1}^{n} p_i \log \frac{p_i}{q_i}$$

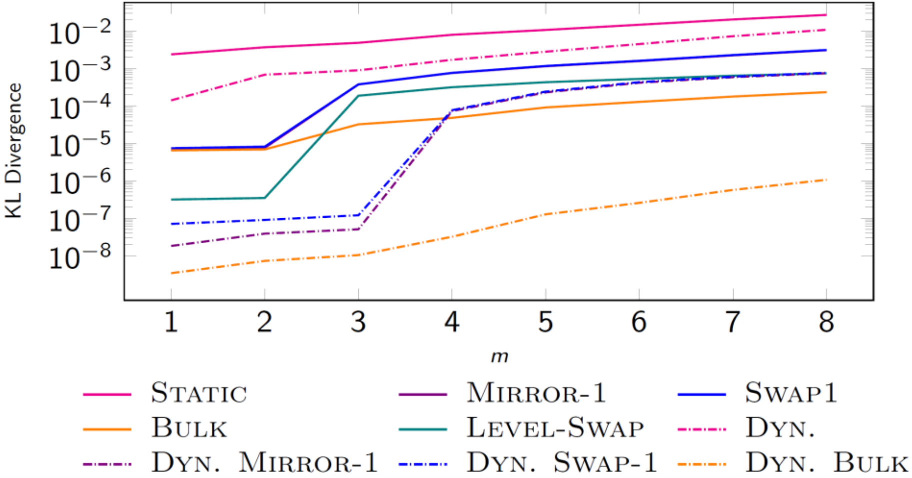
  one-sided, asymmetric, distance from P to Q

- Q is uniform on 2^m elements, $U_m = \{2^{-m}, \ldots, 2^{-m}\}$

- $D_{KL}(P \| U_m) = m - H(P)$,

  $H(P)$ is the entropy of P.

Uniformity: KL distance



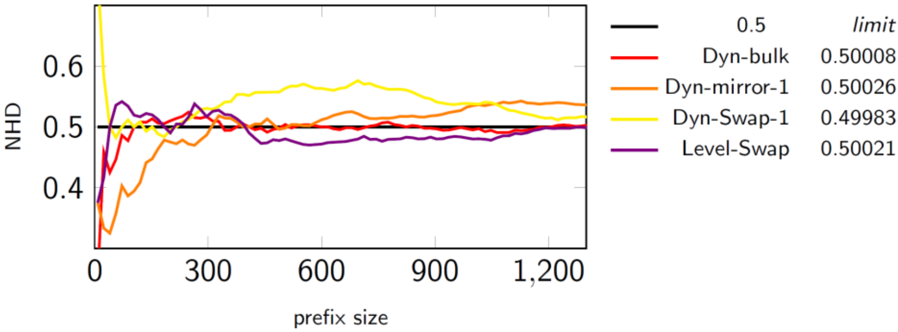
Sensitivity to variations in the secret key.

The Normalized Hamming distance:

Let $A = a_1 \cdots a_n$ and $B = b_1 \cdots b_m$ be two bitstrings and assume $n \geq m$.

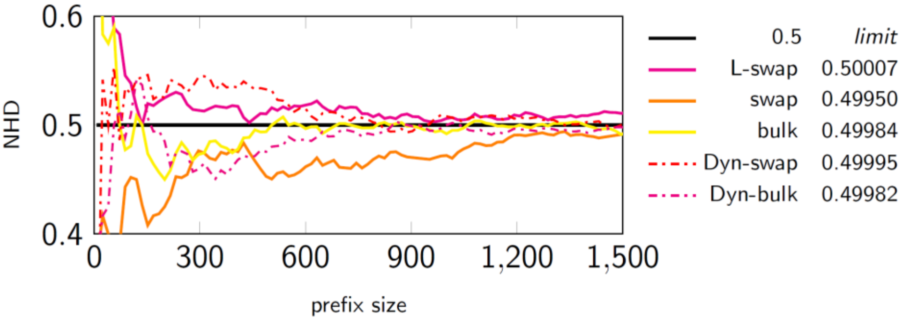
The normalized Hamming distance: $\frac{1}{n} \sum_{i=1}^{n} (a_i \text{ XOR } b_i)$.

Secret Key Variations



NHD between two runs on the same text with different randomly generated keys $\mathcal{K}_1$ and $\mathcal{K}_2$.

CPA security NHD.



NHD for two runs on the same text with the same key, but different DCs.

THANK YOU