ON THE SECURITY OF A SECURE LEMPEL-ZIV-WELCH (LZW) ALGORITHM
Shujun Li1, Chengqin Li2,3 and C.-C. Jay Kuo4
1 University of Konstanz, Germany
2 Xi’an Jiaotong University, Hunan, China
3 The Hong Kong Polytechnic University, Hong Kong SAR, China
4 University of Southern California, USA

Quick Questions and Answers
1. What is this poster about?
Reevaluating the security of a secure LZW algorithm.
2. Who proposed the secure LZW algorithm, when and where?
3. What is your main finding?
The secure LZW algorithm is not sufficiently secure against a chosenplaintext and a chosen-ciphertext attack.
4. How efficient is your chosen-plaintext attack?
Less efficient than the chosen-ciphertext attack, but still manageable.
5. Can the security problems be overcome?
Yes, but at the cost of a higher computational load and/or a lower encoding efficiency.
6. Do you have source code of your attack available somewhere?
http://www.hooklee.com/Papers/ICME2011_SecLZW.zip

Security Re-Evaluation
Two problematic assumptions behind Zhou et al.’s previous security analysis:
- each masking key Ki has to be exhaustively guessed;
- Ki cannot be guessed without guessing all previous keys first. Neither of the two assumptions holds for single-symbol entries!

Theorem 1 Given two different plaintexts S, S′, if S, and S′ are both single-symbol strings, then B = B′ =⇒ S = S′;

Chosen-Plaintext Attack
- Step 1: Choose a number of plaintexts to build a 2-D look-up table (LUT) between all single-symbol strings Si and their ciphertext indices Bi at each position of the plaintext.
- Step 2: For each ciphertext index Bi that can be found in the constructed LUT, output the corresponding single-symbol string Si in the recovered plaintext, otherwise output “*” (an undetermined string).

Lempel-Ziv-Welch (LZW) Encoding
- LZW is a lossless coding scheme based on a dynamic dictionary.
- It is popular because of its use in the UNIX compression tool compress and in the lossless image format GIF.
- The encoding process works as follows:
  1. initialize the dictionary with single-symbol strings of the input alphabet;
  2. find the longest entry W in the dictionary matching the input I;
  3. add the entry index into the output and remove W from I;
  4. add a new entry Wx into the dictionary, where x is the next to-be-encoded symbol (i.e., the first symbol in I);
  5. Go back to Step 2.

Zhou et al.’s Secure LZW Algorithm
- Random insertion of dictionary entries: the index of each dictionary entry is randomized under the control of a keyed hash function.
- Random permutation of dictionary entries: the whole dictionary is organized into a square array, and then permuted columnwise and rowwise under the control of four secret parameters c1, c2, r1, r2.
- Output bitstream masking: masking (encrypting) the output bitstream by XORing it with the keystream generated by a stream cipher.
- Security claims (2n is the dictionary size and L is the plaintext size):
  - security against ciphertext-only attack: (2n);
  - security against chosen-plaintext attack: 2nL.

Possible Enhancements
1. Making the randomization process of dictionary entries and the random permutation process of the dictionary dependent on previously coded symbols. 2. Introducing a session-varying initial vector (IV) that obscure the first single-symbol strings.

Coding Efficiency
- The secure LZW algorithm compromises the coding efficiency by disabling the possibility of using variable-width ciphertext indices.
- A comparison:
  - variable-width LZW encoder: 3356 bits,
  - Zhou et al.’s secure LZW encoder: 3940 bits when n = 10.