Efficient Format-Compliant Encryption of Regular Languages: Block-Based Cycle-Walking

Thomas Stütz, Andreas Uhl

Efficient Format-Compliant Encryption of Regular Languages: Block-Based Cycle-Walking

Thomas Stütz and Andreas Uhl

University of Salzburg,
Department of Computer Sciences
{tstuetz, uhl}@cosy.sbg.ac.at

Abstract. In this work an efficient format-compliant encryption approach for regular languages is proposed, block-based cycle-walking. The approach can transparently trade-off security for efficiency and can be adjusted to the desired level of security. The relationship between the encryption of regular languages and the encryption of JPEG2000 bitstreams is established. Block-based cycle-walking for the regular language of JPEG2000 bitstreams is compared to the extensive previous work from the multimedia security community and to the previous work from the cryptographic community, which is analyzed and evaluated.

1 Introduction

Format-compliant encryption has been intensely researched especially in the multimedia (security) community [11] and recently the cryptographic community contributed to the topic as well [1]. In this work, we define format-compliant encryption as ciphers where both plaintext and ciphertext are in the same format, i.e., meet all the syntactical and semantical requirements of the format definition.

Format-compliant encryption is the key technology to enable security in environments in which security has not been initially incorporated in the design, e.g., PGP can be considered the most widely deployed format-compliant encryption system for the mail format. Other examples are the format-compliant encryption of social security numbers, credit card numbers, 512-byte disk sectors, database entries of a certain type, and multimedia formats, such as JPEG, JPEG2000 and H.264.

For multimedia formats, format-compliant encryption can offer significant application advantages. Modern compression standards, such as JPEG2000 and H.264:SVC, generate a scalable representation of the visual data, that can be efficiently adapted. The preservation of scalability in the encrypted domain allows to perform adaption in the encrypted domain. The preservation of scalability is of fundamental importance for the secure streaming of multimedia content [6]. There have been numerous proposals for scalability-preserving encryption of JPEG2000 (see [3] for a survey), which has also been standardized within JPEG2000 (JPSEC) [8]. Most of the proposed JPEG2000 encryption schemes encrypt only the coded coefficient data (called bitstream in JPEG2000), while

preserving header data. These schemes require encryption schemes which preserve the format of the JPEG2000 bitstream. Our approach is to use the formal description of the JPEG2000 bitstream, i.e, of the regular language describing it, in order to develop a simple yet efficient bitstream-compliant encryption scheme.

Thus in this work we focus on efficient encryption of regular languages, and specifically we show that JPEG2000 bitstreams are a regular language and thus the cryptographically secure format-compliant encryption algorithms (in the sense of the strong notion of message privacy, i.e., MP-security), cycle-walking and RtE (Rank-then-Encipher) [1], can be adapted. We answer the question whether RtE and cycle-walking are computationally efficient enough for practical application. We also propose a new notion of security, MQ security (message quality), which allows to differentiate between weaker and stronger encryption schemes, which are not MP-secure. Block-based cycle-walking is presented, a novel approach. It is the first efficient JPEG2000 encryption approach that does not rely on the availability of an external IV (initialization vector). An extensive analysis and evaluation of its security and complexity is conducted and it is compared to the state of the art.

2 Regular languages

Regular languages have an important role in computer science and applications, specifically in the definition of data formats. In the scope of this work it is most interesting that for regular languages given as DFA (deterministic finite automaton) efficiently computable (linear complexity) ciphers with proven security have been proposed. Regular languages can be described in different ways, e.g., by a DFA accepting the language, or by a regular expression generating the language. A DFA can be formalized as a tuple $(Q, \Sigma, \delta, q_0, F)$; Q is the set of states, Σ is an alphabet, $\delta : Q \times \Sigma \to Q$ is the transition function, q_0 is the start state and F is the set of accept states, following the conventions of [9].

In [7] it is stated that the JPEG2000 bitstream must not contain sequences above $0\,\mathrm{xff8f}$ and must not end with $0\,\mathrm{xff}$. In the following we show that these requirements are equivalent to a regular language, i.e., that all bitstrings obeying the above mentioned syntax requirements can be produced by a regular expression, and are accepted by a DFA.

JPEG2000 Bitstreams Are A Regular Language The regular language of valid JPEG2000 bitstreams B is accepted by a DFA, with only three states, $Q = \{q_1, q_2, q_3\}$, $\Sigma = \{0\mathrm{x}00, \ldots, 0\,\mathrm{xff}\}$, the start state $q_0 = q_1$ and a single accept state $F = \{q_1\}$. The DFA is illustrated in figure 1, from which the transition function δ can be easily derived. The regular expression describing the language of JPEG2000 bitstreams is

$$(0\,\mathrm{xff}(0\mathrm{x}00 - 0\mathrm{x}8\mathrm{f}) \cup (0\mathrm{x}00 - 0\mathrm{xfe}))^*$$

where "()" groups elements to a new regular expression, "$(x - y)$" denotes all the elements between x and y and $*$ denotes 0 and more repetitions of a regular

expression. The equivalence between the syntax requirements for JPEG2000 bitstream and the regular language B can be verified briefly: no string ending with 0xff can be generated as 0xff always yields a non-accept state and if a symbol in excess of 0x8f follows, the DFA goes in a terminal reject state, and thus no sequence in excess 0xff8f is accepted.

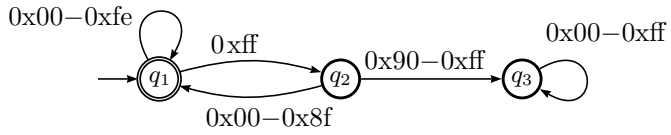


Fig. 1. DFA for B

3 Format-Compliant Encryption

In [1] two secure generic format-compliant encryption schemes, especially for regular languages, are discussed and analyzed.

Cycle-Walking Cycle-Walking encrypts format-compliant messages of a certain length (n bits) and employs an n-bit block cipher for encryption. The plaintext is repeatedly encrypted until the ciphertext is format-compliant. For decryption the ciphertext is decrypted until it is format-compliant. As one decryption step exactly inverts one encryption step, the first format-compliant decryption has to be the plaintext. The scheme is as secure as the underlying n-bit block cipher [2] (see section 4 for a discussion on variable length block ciphers).

Rank-then-Encipher If the format-compliant strings are sparse (in the n-bit domain of the block cipher) the computational complexity of cycle-walking is huge, as it is very improbable that a format-compliant string is produced in the pseudo-random encryption process. In order to reduce the sparsity, ranking of the language L can be employed. Ranking assigns each string of L with length n an unique number, its index, and the indices are consecutive natural numbers in the range of 1 to $|L_n|$, the number of strings of L with length n. The index can then be encrypted with the cycle-walking technique and a variable-length block cipher, which reduces the expected number of necessary iterations i for cycle-walking ($\mathrm{E}(i) \leq 2$).

In [1] it is pointed out that regular languages can be efficiently ranked and algorithms are given.

JPEG2000 Bitstream Encryption For the JPEG2000 bitstream numerous algorithms have been proposed, an overview is given in [3]. A variation of the cycle-walking technique has been proposed for JPEG2000 bitstreams[13]. Apart from cycle-walking, all proposed length-preserving schemes preserve to a certain extent plaintext bits and are thus not secure with respect to the strong cryptographic security notion of MP-security (see section 5.1). An **exemplary efficient algorithm** [12] preserves two-byte-sequences starting with 0xff and encrypts the remaining bytes in the range 0x00−0xfe. In figure 1, this corresponds to encrypting only values in the loop of state q_1, a simple explanation why format-compliance is preserved. While the format-compliance of this algorithm is easily shown, for more complex algorithms this is quite a tedious task and several flawed algorithms have been proposed. Also the amount of preserved plaintext data is tedious to assess for more complex algorithms.

4 Block-Based Cycle-Walking

Block-based cycle walking is a simple, efficient algorithm for the format-compliant encryption of regular languages. Security can be transparently traded off against efficiency, by choosing the appropriate block size. The approach depends on the specifics of the language.

The encryption algorithm splits an arbitrary length plaintext string into blocks. Each block consists of symbols, which determine a sequence of states of the describing DFA. These blocks are processed in order. Each block is encrypted repeatedly until it is format-compliant and the last state of the state sequence of the ciphertext is the same as of the state sequence of the plaintext. In the determination of format compliance the previous block's end state has to be considered, i.e., the DFA starts in the previous block's end state.

The decryption algorithm splits the ciphertext into blocks. Each block is repeatedly decrypted until it is format-compliant and its end state is the same of the ciphertext block (which of course is the same as of the plaintext block). In the determination of format compliance the previous block's end state has to be considered as well. Thus for every block concise conditions for the determination of the necessary iterations are available and the cycle-walking technique can employed on each block.

The scheme elegantly reduces the complexity of cycle-walking of the entire string to the complexity of cycle-walking small blocks. The security breach is limited to the preservation of a single state per block, which has different implications on the security for different regular languages.

In the following we discuss the approach for JPEG2000 bitstreams in detail.

Algorithm Description The format-compliant encryption of a JPEG2000 bitstream is given in listing 1.1, parameters are a string of bytes, which has to be format-compliant, the key and the block length l. The bitstream is split into blocks and each block is encrypted with the function fc_encblock which additionally gets the last byte of the previous block which is used in the preservation

Listing 1.1. Format-compliant encryption

```
fc_bitstream fcencrypt( fc_bitstream f,
        key k, block_length l)
{
        plb = 0;
        t = 0;

        while( more_blocks( f, l ) )
        {
                block = get_block( f, l );
                lb = last_byte_of( block );
                fc_encblock( block, plb, k, t++ );
                plb = lb;
        }
        return f;
}
```

of format-compliance. The function get_block(f, l) returns the next block of length l, except for the last two blocks (in order to cope with the minimal block size of the underlying cipher). Internally in get_block, the bitstream is split into blocks of length l the last block can have arbitrary length $\leq$ l. Each block is returned except for the last two blocks, for which the behaviour depends on the minimal block length m the underlying cipher can handle: if the length of the last block is below m, the preceding block is returned with the last block appended. The function more_blocks implements this behaviour.

The encryption of a block is given in listing 1.2. Blocks are simply encrypted until they are valid, i.e., containing no sequence in excess of 0xff8f, the first byte must be below 0x8f if the previous byte is a 0xff byte, and for the last cipher byte must preserve whether the byte was 0xff (see listing 1.3). The function enc_vblock calls an variable length block cipher with minimal length m.

The format-compliant decryption works exactly the same, except that the decryption function of the variable length cipher is called.

Requirements The algorithm relies on the availability of an arbitrary length cipher. An arbitrary length tweakable block-cipher would be optimal. Having a secure block cipher with a fixed block size, e.g., AES with 128 bit, secure block-ciphers with a larger block size can be constructed [1]. Currently there are no secure constructions for smaller block size ciphers known [2]. Thus a BBCW block length smaller than the block size of the employed block cipher is not directly supported, solutions are padding or method 1 of [2]. If the last block is smaller than the cipher block size it can be encrypted jointly with the preceding block, as block ciphers with a larger block size can be constructed.

5 Security Analysis and Evaluation

In order to analyze and compare the security of block-based cycle walking, we first discuss sensible notions of security for format-compliant encryption.

Listing 1.2. Format-compliant encryption of a block

```
fc_block fcenc_blk( fc_block b,
        last_byte_of_previous_block plb,
        key k, tweak t )
{
        lb = last_byte_of( b );
        while ( !is_valid( b, plb, lb ))
        {
                enc_vblock( b, k, t );
        }
        return b;
}
```

Listing 1.3. Checking the validity of a block

```
bool is_valid( fc_block b,
        last_byte_of_previous_block plb,
        last_byte_of_current_block lb )
{
        bsc = contains_seq_in_excess_of_0xff8f( b );
        bprevblk = (plb != 0xff) ||
                        (first_byte_of( b ) <= 0x8f);
        if (lb == 0xff)
                blb = last_byte_of( b ) == 0xff;
        else
                blb = last_byte_of( b ) != 0xff;

        return  bsc && bprevblk && blb;
}
```

5.1 Security notions

A security notion defines the precise meaning of the term security. We adopt the framework of [1], but refrain from most technical details here. In [1] the security notions, MP security and MR security are formalized as code-based games, the notion of MQ security is introduced in this work.

MP security For MP security (message privacy) an adversary must not be able to compute any property of plaintext from the ciphertext. E.g., an encryption scheme preserving the digit sum of the plaintext is not secure in the sense of MP security.

MR security For MR security (message recovery) an adversary must not be able to compute the plaintext from the ciphertext. E.g., an encryption scheme preserving the digit sum of the plaintext can be perfectly secure in the sense of MR security.

We think that for the majority of applications MR security is too weak, while MP security is too strong. A good example is image encryption: an MR-secure encryption scheme may still give an adversary access to a visually indistinguishable high-quality approximation of the plaintext, while a perfect recovery of the plaintext is infeasible (consider LSB encryption). MP security is, however, not the security goal as e.g. the preservation of checksums in the encrypted domain could even be considered a beneficial feature (error correction could be done on the ciphertext!). A similar rationale is necessary for format-compliant encryption: provably MP-secure encryption is often practically infeasible (see section

6), but we still want to distinguish "really insecure" schemes, that encrypt only a small fraction of the plaintext, from "rather secure ones". We want to define a security notion that is capable to distinguish between a scheme which only encrypts a block of the plaintext (as MR-secure as the block cipher, e.g., AES) and a scheme which only preserves few bits, commonly to ensure format-compliance.

MQ security In MQ security (message quality), security is defined by the inability of an adversary to compute an approximation of the plaintext with higher "quality" than targeted. The quality needs to be measured, for the encryption of regular languages we propose the nHd (normalized Hamming distance) of the binary representation of the strings of the regular language as security metric. Thus schemes that only encrypt a small fraction are insecure (nHd ≈ 0). Other security metrics, modeling the actual security requirements of an application, can be employed in the MQ security notion (e.g., the MSE for images). MQ-secure schemes should yield a nHd of plain- and ciphertext that is close to $\mathrm{E(nHd)}_L$, the expected nHd of two randomly-chosen, equal-length strings of the language L. An adversary is successful in breaking a scheme in MQ security if she computes an approximation that is closer to the original as defined in the security definition, i.e., if she can compute a reconstruction Y with an nHd to X that differs from the expected (the security level can be adjusted by ϵ): $|\mathrm{nHd}(X, Y) - \mathrm{E(nHd)}_L| < \epsilon$.

MQ-security with the nHd as metric is rather similar to defining a certain encryption ratio as secure. The encryption ratio, however, is hard to assess if we consider the encryption process a black box (the encryption ratio of some schemes depends on the plaintext and the key), which randomly preserves some of the plaintext data.

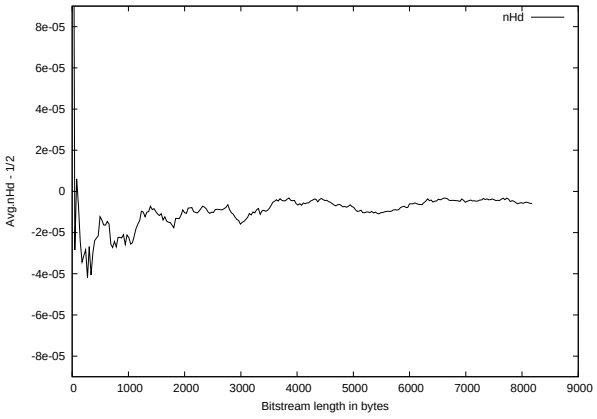


Fig. 2. Average nHd-1/2 of 10.000 random string pairs of B

In the case of arbitrary bitstrings ($A = \{0,1\}^*$) and a uniform distribution of the plaintexts, $\mathrm{E(nHd)}_A$, the expected nHd of two equal-length random strings is $1/2$, for other languages L the $\mathrm{E(nHd)}_L$ can differ. In the following we investigate the $\mathrm{E(nHd)}_B$ for the regular language of JPEG2000 bitstreams B. On average the nHd of two randomly chosen strings of B (uniform distribution), which is an estimate of $\mathrm{E(nHd)}_B$, is very close to $1/2$, as can be seen in figure 2.

Relations between notions MP security implies MQ security, which implies MR security for sensible choices of d and ϵ.

$$\mathrm{MP} \Rightarrow \mathrm{MQ}_{\mathrm{nHd},\epsilon>0} \Rightarrow \mathrm{MR}$$

The implications work only in one direction. Thus an MQ-secure scheme needs not to be MP-secure.

Analysis Of the discussed format-compliant encryption algorithms, only cycle-walking [2] and RtE + cycle-walking achieve MP-security. Block-based cycle-walking only preserves data if the last byte of a block equals 0xff. Thus in the worst case every block's last byte 0xff byte is preserved, on average every $1/256$ block's last byte is a 0xff byte, while the best case is that no byte is preserved. Thus the desired level of MQ security can simply be achieved by choosing the appropriate block size. For the exemplary efficient algorithm (see section 3) the worst case is that every 2^{nd} byte is an 0xff byte and thus the entire string is preserved (MR insecure!), in average every 256 byte equals 0xff and in the best case no 0xff byte is present.

Evaluation The security of block-based cycle-walking is experimentally compared to previously proposed JPEG2000 bitstream encryption algorithms. Note that the experimental evaluation of security only gives an upper-bound for MQ-security, i.e., schemes evaluated as insecure are insecure, schemes evaluated as "secure" may have hidden flaws. A simple example is an "encryption" scheme which inverts the first half of the bits and preserves the other, the average nHd would be exactly $1/2$, while an adversary can easily compute a reconstruction with nHd zero, by inverting the first half again (though MQ-insecure). The average nHd of 10.000 plaintext ciphertext pairs for each length is illustrated in figure 3. Block-based cycle-walking with block size b is labeled "BBCW x", the exemplary efficient encryption algorithm (see section 3) is labelled "Wu Ma". The other algorithms follow the nomenclature in [3]. Of all evaluated schemes (each scheme discussed in [3] has been evaluated) only the approach of [4] achieves a comparable MQ-security level very close to $1/2$, the expected nHd of two random strings of B, the regular language of JPEG2000 bitstreams. For a very large string length $n = 2^{20}$ results for selected algorithms are given in table 1.

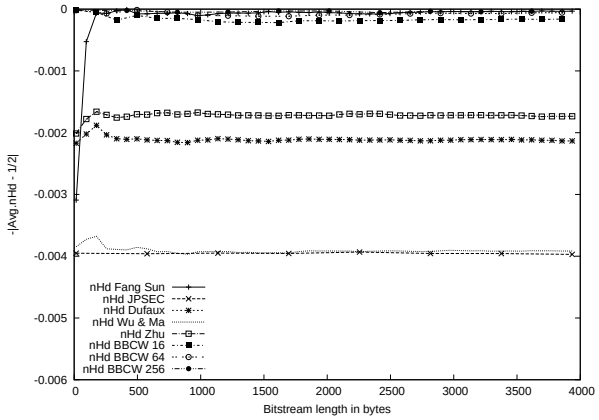


Fig. 3. Average normalized Hamming distance for different encryption approaches

Fang	JPSEC	Kiya	Kiya10	Dufaux	WuBlk	Zhu	bbcw16	bbcw128	bbcw256	AES
.50009	.49606	.23444	.02344	.49787	.49611	.49830	.50001	.49999	.50000	.50001

Table 1. Average nHd for 1MB and 100 trials

6 Complexity Analysis and Evaluation

The runtime performance of block-based cycle-walking (in dependency of the block size), other JPEG2000 bitstream encryption algorithms, cycle-walking and RtE is compared, analyzed and evaluated. An interesting question is whether RtE performs better than cycle-walking, the theoretical analysis is in favour (if the format-compliant strings are sparse), but the increased complexity of multi-precision arithmetic is not considered in this analysis.

6.1 Analysis

The runtime complexity of cycle-walking for JPEG2000 bitstreams has been shown to increase exponentially with the string length n ($\mathcal{O}(2^n)$) [10]. The space complexity grows linearly, i.e., is in $\mathcal{O}(n)$.

The ranking of a string of a regular language consists of two steps: build the table T and rank (requires T). The table needs to be built only once for several ranking operations of same length strings. Building the table has complexity $\mathcal{O}(n|Q||\Sigma|)$, and the table is of size $\mathcal{O}(n|Q||\Sigma|\log|B_n|)$, where $|B_n|$ is the number of strings with length n. The runtime complexity of ranking is $\mathcal{O}(|\Sigma|n)$ and the space complexity is constant, but T is required. The arithmetic, however, has to be performed on numbers which correlate in size with $|B_n|$.

The runtime of block-based cycle-walking is linear in the length of string and for the block size b the complexity is the same as that of cycle-walking, which is efficient for short JPEG2000 bitstreams (below 0.001s for $b < 1500$ bytes,

see figure 5(a)). The space complexity is constant, as the bitstream can be read block by block.

Most of the JPEG2000 bitstream encryption approaches discussed in [3] have a runtime linear in n and constant space complexity, i.e., are capable of stream processing.

6.2 Evaluation

The experiments have been conducted on a Intel Core2 6700@2.66GHz, using custom C implementations which employ GMP 4 (gnu multi-precision library). The source code of the implementations and the experiments will be available at www.wavelab.at. As secure encryption primitive AES has been employed, the variable length cipher is mimicked by AES in ECB mode with ciphertext stealing, which may result in too optimistic results for cycle-walking (but cycle walking is infeasible even with this efficient implementation).

Cycle-walking, RtE The actual runtime complexity of ranking is not linear, the multi-precision arithmetic takes its toll (see figure 4). The size of the table T (the number of entries of T is linear in n) is illustrated in figure 4. The size of its entries correlates with $\log |B_n|$ and $|B_n|$ increases exponentially with n [10]. Plain cycle-walking is way more efficient than RtE (compare figure 4 and figure 5(a)). However, for larger string length n the complexity of cycle-walking increases dramatically as well.

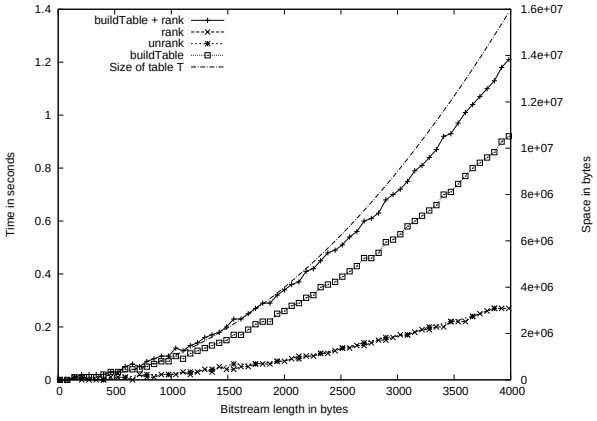
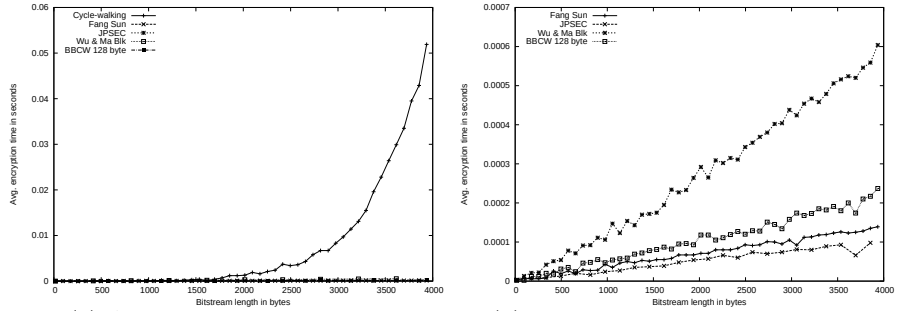


Fig. 4. Space and time of RtE

Efficient JPEG2000 Bitstream Encryption The results have been obtained by averaging 10.000 independent trials. The runtime performance of cycle-walking and efficient JPEG2000 encryption algorithms is illustrated in figure



(a) All encryption algorithms (b) Efficient encryption algorithms

5(a). The average encryption time of cycle-walking increases exponentially with increasing string length, which greatly reduces the applicability of cycle-walking for JPEG2000 encryption [10]. In figure 5(b) the runtime performance of efficient JPEG2000 bitstream encryption approaches is compared in detail. The performance of block-based cycle-walking with different block sizes has been evaluated for block sizes from 16 byte to 256 byte. The performance increases with the block size up to 128 byte, than the effect reverses (these effects are highly implementation and hardware dependent). For a very long string of 1MB, i.e., $n = 2^{20}$ results are given in table 2. In summary block-based cycle-walking is well-performing.

Fang	JPSEC	Kiya	Kiya10	Dufaux	WuBlk	Zhu	bbcw16	bbcw128	bbcw256	AES
.0359	.0263	.0266	.0068	.0354	.0364	.0340	.1030	.0573	.0657	.0235

Table 2. Average time (seconds) for 1MB and 100 trials

7 Conclusion

An efficient format-compliant encryption approach for regular languages has been proposed, namely block-based cycle walking. The general approach has been implemented for the regular language of JPEG2000 bitstreams. The regular language of JPEG2000 bitstreams is dense enough that ranking is inefficient and sparse enough that cycle-walking is inefficient. Thus MP-secure encryption algorithms are practically infeasible. For JPEG2000-bitstreams, block-based cycle-walking greatly reduces the complexity compared to cycle-walking, from exponential time and linear space to linear time and constant space (in plaintext length). Compared to previous JPEG2000 encryption schemes it is simple and elegant (its reversibility is easily shown) and offers increased security.

References

1. M. Bellare and T. Ristenpart. Format-preserving encryption. In *Proceedings of Selected Areas in Cryptography, SAC '09*, Calgary, Canada, Aug. 2009. Springer-Verlag.
2. J. Black and P. Rogaway. Ciphers with arbitrary finite domains. In *Topics in Cryptology CT-RSA 2002*, volume 2271 of *LNCS*, pages 185–203. Springer-Verlag, Jan. 2002.
3. D. Engel, T. Stütz, and A. Uhl. A survey on JPEG2000 encryption. *Multimedia Systems*, 15(4):243–270, 2009.
4. J. Fang and J. Sun. Compliant encryption scheme for JPEG2000 image code streams. *Journal of Electronic Imaging*, 15(4), 2006.
5. O. Goldreich. *The Foundations of Cryptography*. Cambridge University Press, 2001.
6. H. Hellwagner, R. Kuschnig, T. Stütz, and A. Uhl. Efficient in-network adaptation of encrypted H.264/SVC content. *Elsevier Journal on Signal Processing: Image Communication*, 24(9):740 – 758, July 2009.
7. ISO/IEC 15444-1. Information technology – JPEG2000 image coding system, Part 1: Core coding system, Dec. 2000.
8. ISO/IEC 15444-8. Information technology – JPEG2000 image coding system, Part 8: Secure JPEG2000, Apr. 2007.
9. M. Sipser. *Introduction to the Theory of Computation*. International Thomson Publishing, 1996.
10. T. Stütz and A. Uhl. On format-compliant iterative encryption of JPEG2000. In *Proceedings of the Eighth IEEE International Symposium on Multimedia (ISM'06)*, pages 985–990, San Diego, CA, USA, Dec. 2006. IEEE Computer Society.
11. A. Uhl and A. Pommer. *Image and Video Encryption. From Digital Rights Management to Secured Personal Communication*, volume 15 of *Advances in Information Security*. Springer-Verlag, 2005.
12. H. Wu and D. Ma. Efficient and secure encryption schemes for JPEG2000. In *Proceedings of the 2004 International Conference on Acoustics, Speech and Signal Processing (ICASSP 2004)*, pages 869–872, May 2004.
13. Y. Wu and R. H. Deng. Compliant encryption of JPEG2000 codestreams. In *Proceedings of the IEEE International Conference on Image Processing (ICIP'04)*, Singapure, Oct. 2004. IEEE Signal Processing Society.