An Approach to Achieve Compression Along with Security via a User Assigned Security Key with Possible Lossless Compression

Ashis Datta and Mousam Saikia

Abstract Although the current version of this research work might not be fool-proof as this kind of works are currently being researched upon and yet to be successfully implemented at large. Popular software such as WinRAR makes use of AES to encrypt the data and also compresses it at the same time but only one algorithm is used to compress the data [1]. It is sometimes seen that using different lossless compression algorithms multiple times can achieve a better compression ratio and may even achieve better security [2]. Our plan is to achieve such kind of compression ratios while achieving confidentiality with a model that we have devised ourselves.

Keywords Data compression · Loss-less · Encryption · Decryption · Cipher · Private key · Symmetric key · Plain text · Cryptographic modeling · Digital right

1 Introduction

1.1 Encryption

The process of converting from plaintext to cipher-text is known as enciphering or encryption. Some classical encryption techniques [3] include symmetric cypher model, substitution techniques, transposition techniques, rotor machines and steganography. Our algorithm partially falls into the category of symmetric cypher model. In this model, the sender encrypts the plain text into some cypher text using some key to encrypt the data. The same key has to be used to decrypt the data. Our algorithm modifies this concept by making sure that the original content that is sent by the sender can be decrypted only in a specific system i.e., of the receiver. We achieve this by incorporating the receiver's system's MAC address into the final key that is used to encrypt data.

A. Datta (✉) · M. Saikia
Sikkim Manipal Institute of Technology, Majitar, Sikkim 737136, India
e-mail: ashis.d@smit.smu.edu.in

H. K. D. Sarma et al. (eds.), *Trends in Communication, Cloud, and Big Data*, Lecture Notes in Networks and Systems 99, https://doi.org/10.1007/978-981-15-1624-5_12

1.2 Compression

There are two basic types of data compression [4, 5]; lossy data compression and lossless [6] data compression.

In lossy data compression techniques, the original data upon decompression will lose or undergo changes in some of its minute features which are not easily noticeable by humans. This type of compression technique is used in compression of images, audio and video data.

Lossless compression techniques are those in which, upon decompression will produce an exact replica of the original data. Our software makes use of this type of algorithms to achieve compression. The technique we have used should be able to compress general-purpose data, i.e. the algorithm will accept any bit string given to it.

We present a new algorithm for lossless data compression and encryption. The basic idea of the algorithm is to define a unique compression and encryption of files specified by a user based on a key/password provided by the user. We are making use of lossless data compression [7] algorithms such as Snappy, Gzip, BZip2 and XZ. The key specified by the user is taken as an input and files given are compressed multiple times depending upon the result of XOR operation between the password and the MAC (Media Access Control) address of the user's system which is automatically retrieved by the software and is unique to each system.

During the process of decryption and decompression of the data, the contents will be available at a temporary sandboxed location in a folder and a process will be created that will keep on running till its closed or the system is turned off. The extracted file(s) will remain secure as copying, distributing or tampering with it will not be possible. This is achieved by running the decryption/decompression software as non-administrator. This part of the software will terminate itself if it is being ran with administrator privileges. The contents will be deleted if the process is killed and the space in the hard drive the content takes up will be filled with random data such that a hacker cannot retrieve that data later. This will be achieved by replacing the content of the data just before deleting them.

1.3 The Problem of Piracy

The way we consume media is changing and as content owners and producers look for new ways to keep audiences engaged, the challenges of protecting content across multiple screens becomes increasingly difficult. Innovation, of course, is a good thing, but as new ways of streaming content are developed, it is actually aiding digital piracy.

Piracy is detrimental to innovation, directly affecting job creation and economic growth. Industries protect their ideas through a variety of legal instruments such as patents, copyrights, designs, models and trademarks. Without the protection of their

intellectual property rights [8], they may be less inclined to develop new ideas and products. Risks are particularly high for industries in which the research and development costs are high compared to the production costs of the finalised product. Faced with a diminishing turnover due to counterfeiting and piracy, industry investment in research and innovation could well slow down. This would limit development, growth and competitiveness, forcing industries to simply close or at least limit production.

The problem of piracy is that it's here to stay and it could be argued that the problem is growing; in August it was reported that Game of Thrones was the world's most pirated TV show, with 1.6 million illegal downloads in just four weeks. This accounted for over a quarter of all pirated downloads from the top 100 torrent sites. Another recent report from Viacess-Orca highlighted that during the last Football World Cup they monitored 20 million illegal downloads. What this demonstrates is that the pirates appear to be winning. However the definition of piracy encompasses many different layers: from the amateurish camera recordings shot discreetly from a cinema, through to Internet-streamed content that has been cracked and then uploaded to an app store.

The many layers of piracy certainly means that content owners and producers can never fully protect their content, they can only do as much as possible to limit what pirates can access, and this is where it gets interesting. To implement a DRM (digital rights management) solution for streamed content across all devices is both complicated and expensive, so what we're seeing is that studios are choosing to focus their protections around high value digital media such as High Definition and 4 K over standard definition content. What they need is content protection that follows the content and therefore is device agnostic.

1.4 DRM (Digital Rights Management)

Implementing an effective DRM security is the answer and as most content publishers will argue, it is vital to facilitate continued innovation in digital media. As new devices that stream apps and host media players are constantly being developed, the need for secure and unobtrusive digital distribution is urgent. In order for DRM solutions [9, 10] to work to protect the sale of books, films, and music that is growing online, the critical component of a digital key, which allows a user or device to decode the protected content - is required [11]. However, even the best encryption schemes are useless if a hacker can quickly acquire the key.

The costs of DRM security [12] breaches are significant; with the Motion Picture Association of America (MPAA) estimating it costs the film industry $6 billion per year in lost revenue. The weakness with DRM security and implementation is that it can be easily hacked and cumbersome to users if strong application-level defences are not leveraged. In an attempt to curb piracy, many DRM vendors are resorting to using invasive digital rights protection techniques that assume the system is always under attack by pirates, which causes restrictions and performance degradation to honest users.

Since most streaming media applications are performance intensive, DRM security solutions mustn't noticeably impact performance; hence it's not an option. For a DRM system to be well received, it's imperative that the original content simply looks and works better than the pirated copy. The solution lies in imposing DRM security strategies that are effective at preventing piracy, while not degrading the consumer's experience.

DRM vendors are always going to be held hostage to the speed of technological innovation. The problem here, is that the quality of the very content that we on our different devices is likely to be undermined by the massive loss of revenue caused by piracy.

2 The Scheme

The algorithm has two modules with two main parameters each. The modules are briefly described below:

The file processor: This module is responsible for the generation of a special form of file that is both an encrypted and compressed version of a user assigned file.

The reverser: This module is responsible for producing the exact replica of the original file on acquiring the proper file and key (the file that was processed with the key that was assigned during the execution of the previous module).

Both the modules have one common parameter i.e., the key assigned by the user, which needs to be entered during separate instances. The first instance being the processing of an unprocessed file and the second instance being the decryption and decompression of a processed file. During the processing of an unprocessed file the second parameter will contain the path and name of the unprocessed file in the form of a string. While obtaining the original file, the second parameter will contain the path and name of the previously processed file also in the form of a single string. The programming language used for the development of this project is java.

3 Literature Survey

Serial number	Author name	Paper title	Features
1	Robert Franceschini and Amar Mukherjee	Data compression using encrypted text	• It establishes that encrypted representation of text leads to substantial saving of storage space • It is an interesting dictionary based compression method with better performance than gzip and arithmetic coding

(continued)

(continued)

Serial number	Author name	Paper title	Features
2	Matt Mahoney	Data compression explained	• Covers a lot about lossless data compression algorithms and commonly used lossy data compression algorithms • Most of the lossless data compression algorithms are used in this project
3	Mark Stamp	Digital rights management: the technology behind the hype	• Covers topics such as DRM tethered and untethered systems • Covers media snap DRM in detail
4	Chung-Ping Wu	Design of integrated multimedia compression and encryption systems	• It was shown that security could be achieved without sacrificing the compression performance or the processing speed
5	Cappaert, Nessim Kisserli, Dries Schellekens and Bart Preneel	Self-encrypting code to protect against analysis and tampering	• Covers topics such as code obfuscation and self-modifying code • Shows how two or more files are dependent upon one another and achieves self-modifying code
6	M. VidyaSagar, J.S. Rose Victor	Modified run length encoding scheme for high data compression rate	• Covers the run length encoding scheme in details and how the modified version is better from other algorithms in terms of both compression and performance. But better compression algorithms are available
7	Alan Story	Intellectual property and computer software	• Shows how proprietary software can be prevented from piracy • Shows the various ongoing problems for such software • Shows how to tackle these kind of problems

(continued)

(continued)

Serial number	Author name	Paper title	Features
8	Kun-Won Jang, Chan-Kil Park, Jung-Jae Kim and Moon-Seog Jun	A study on DRM system for on/off line key authentication	• This paper shows an algorithm that can encrypt files by dividing it into blocks. Conventional systems can replay digital content only after entirely decrypting it • It can be used to provide free demos for users
9	Pasi Tyrväinen, Jarmo Järvi, Eetu Luoma	Peer-to-peer marketing for content products combining digital rights management and multilevel marketing	• It states "out of the $32 billion market for music in 2002 only about $0.09 billion is sold by paid downloads and online retailers accounted for a mere 1% of music sales" • Shows a model for distribution of copyrighted content in a peer-to-peer network
10	Mikko Löytynoja, Tapio Seppänen, Nedeljko Cvejic	Experimental DRM architecture using watermarking and PKI	• The paper describes a unique architecture for DRM and discusses watermarking in detail

3.1 Objective of This Paper

The primary objective of the paper is to develop a unique algorithm for compression and encryption of files specified by a user based on a key/password provided by the user. We are making use of lossless data compression algorithms such as Snappy, Gzip, BZip2 and XZ. A key comprising of integers that has a variable length from 1 to 19 digits specified by the user is taken as an input and files given are compressed multiple times depending upon the result of XOR operation between the password and the MAC address of the user's system which is automatically retrieved.

It is intended to be used in industries where piracy causes huge amounts of loss. With more research into this field, copyright laws will benefit a great deal in the future. It also has the possibility to become the norm for downloading and uploading of copyrighted content from the internet.

3.2 Solution Strategy

We propose a solution strategy by applying the following steps:

- By continuously running an algorithm to clear the clipboard as a thread in the background. This thread is stopped when the program is closed.
- Upon exit, the files previously extracted to a temporary location will be forcibly deleted.
- The user won't have write access to the files as modifying or coping the contents of these files will make this project useless.
- The key passed by the user will undergo XOR operation with his/her MAC address which is unique for every computer and the result will be used for the encryption instead of the actual key.

4 An Implementation

4.1 Architecture Diagram

In Fig. 1,

T Represents the original unprocessed file.
C Represents the first module.
k Represents the key.
*T Represents the file after processing.

In Fig. 2,

*T Represents the processed (encrypted and compressed) file.

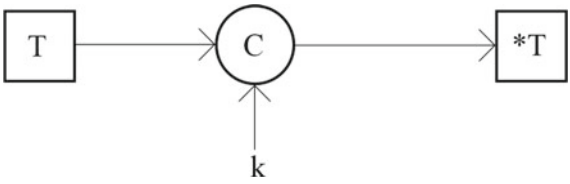

Fig. 1 Block diagram for the first module (compresses and encrypts)

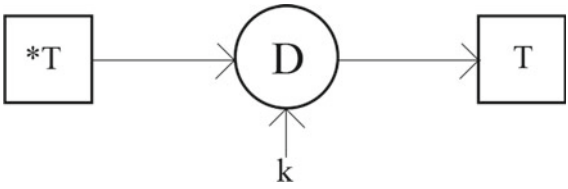

Fig. 2 Block diagram for the second module (decompresses and decrypts)

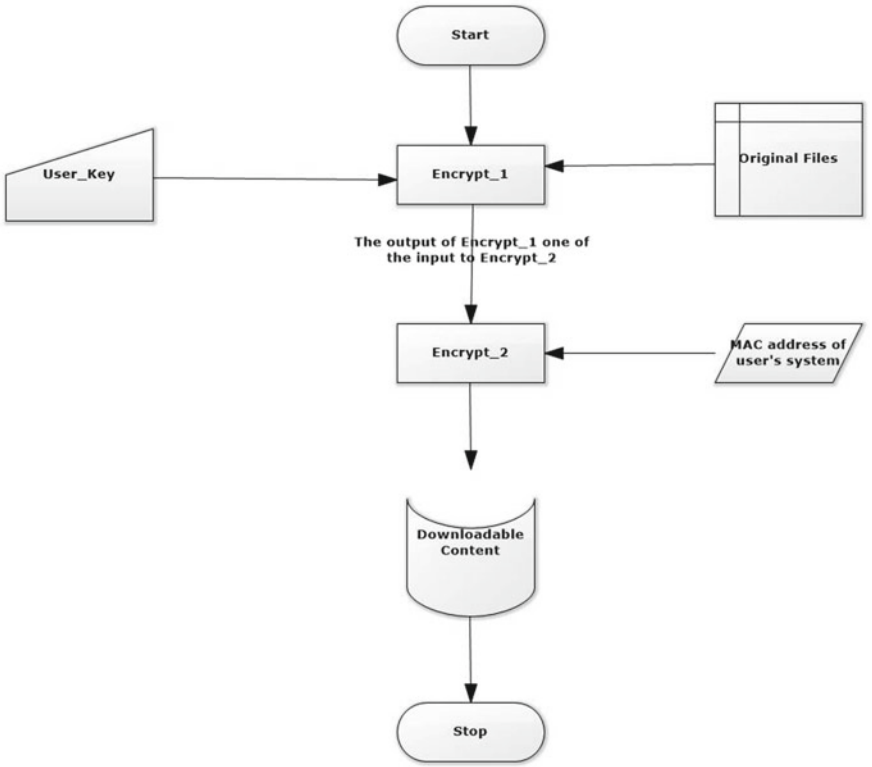

Fig. 3 The file processor/the first module

D Represents the reverser module.
k Represents the key.
T Represents an original copy of the unprocessed file.

4.2 Detailed Diagram

See Fig. 3.

4.3 Detail Process and Pseudo Codes

Compression of data in a lossless manner by making use of various algorithms more than once sometimes achieve better compression ratios. But this kind of algorithms are currently looked into and researched upon without any assurance that it is always

possible. However, the algorithms we are using are quite simple and easy to use. The source codes of these algorithms are easily available on the internet. Snappy is a fast compression and decompression technique achieving speeds of 250 MB/s for compression and 500 MB/s for decompression on a single core i7 of 64 bit. It is based on LZ77 algorithm which achieve compression by replacing repeated occurrences of data with references to a single copy of that data existing earlier in the uncompressed data stream. XZ compression algorithm is a dictionary based compression technique. Gzip makes use of one of the algorithms already defined within itself depending upon the type of data it is given to compress. BZip2 makes use of Burrows-Wheeler algorithm to compress data. Our algorithm can compress multiple files but it is not an archival tool. Although Gzip is an archival tool, this algorithm will not make use of the ability because the file(s) after decryption and decompression will be found at a temporary location in a temporary folder. The software will be fairly easy to use. First, the user needs to specify the file(s) to be compressed and then in the second step, a password is asked to be entered by the user and the system's MAC address is retrieved which then undergoes XOR operation with the password provided. One assumption of our algorithm is that the password entered will be of numbers only with a maximum length of nineteen digits and a minimum length of one digit. The file(s) then undergo lossless compression via various algorithms according to the result generated during the XOR operation. BZip 2, Snappy, Gzip, XZ are lossless data compression algorithms [13] used in building this software. Their algorithms are described below. However, in these algorithms, a file is created and the data output stream is written onto it. In our algorithm, these functions instead of creating an entire file, will return the output stream which will be further compressed. The output file will be created only when it has finished compressing that stream. The resulting stream will also be encrypted.

The pseudo code for BZip 2 compression algorithm in java:

Get the original file.
Create an empty output file.
Initialize output stream.
Read content of the original file.
Initialize BZip2 output stream.
Apply Burrows-Wheeler transform on the data and compress.
Write data to the stream.
Write contents of the stream to output file.
Close output stream.

The pseudo code for BZip 2 decompression algorithm in java:

Get compressed file.
Store contents of compressed file in byte array.
Initialize integer variable n to 0.
While (Reading of data in input stream is not complete, set n as counter)
{

```
   Decompress the contents of the input stream.
   Write contents to the output buffer.
}
Close all input and output streams.
```

The pseudo code for Snappy compression algorithm in java:

```
Enter file to be compressed.
Initialize input stream to be compressed from the input file.
Initialize the output file.
Initialize the output stream.
Initialize encoder.
Apply Snappy algorithm.
Write the output to the output stream.
Create the output file.
Close output stream.
```

The pseudo code for Snappy decompression algorithm in java:

```
Read contents of compressed input file.
Put the content in a byte buffer.
Initialize Snappy decoder.
Decompress the contents by setting the correct properties.
Put decompressed content in an output stream.
```

The pseudo code for Gzip compression algorithm in java:

```
Initialize input stream.
Initialize the output stream.
While (Reading of data in input stream is not complete)
{
   Apply Gzip algorithm to data.
   Write to the output stream.
}
Close output stream.
```

The pseudo code for Gzip decompression algorithm in java:

```
Get compressed file.
Store contents of compressed file in byte array.
Initialize integer variable read_bytes to 0.
While (Reading of data in input stream is not complete, set read_bytes as counter)
{
   Decompress the contents of the input stream.
   Write contents to the output buffer.
```

```
}
Close all input and output streams.
```

The pseudo code for XZ compression algorithm in java:

```
Initialize input stream.
Initialize output stream.
Set size of dictionary.
While (Reading of data in input stream is not complete)
{
   Apply XZ Algorithm.
   Write compressed contents to output stream
}
Close output stream.
```

The pseudo code for XZ decompression algorithm in java:

```
Get compressed file.
Store contents of compressed file in byte array.
Initialize integer variable read_bytes to 0.
While (Reading of data in input stream is not complete, set read_bytes as counter)
{
   Decompress the contents of the input stream.
   Write contents to the output buffer.
}
Close all input and output streams.
```

The code for function to detect whether the program is being run as administrator:

```
Initialize a Boolean flag.
Initialize a string with the command reg query "HKU\S-1-5-19"
Run the command in command prompt.
If no error is found set the flag to false.
Else set the flag to true.
```

A pseudo-code for one round of the encryption process

```
int key_1, key_2, secret_1, pass_value;
if(key_1 is not divisible by secret_1 and key_2 is not divisible by secret_1)
   use BZip 2 on file_1;
if(key_1 is not divisible by secret_1 and key_2 is divisible by secret_1)
   use Snappy on file_1;
if(key_1 is divisible by secret_1 and key_2 is not divisible by secret_1)
```

```
    use GZip on file_1;
if(key_1 is divisible by secret_1 and key_2 is divisible by secret_1)
    use XZ on file_1;
pass_value = XOR(key_1, key_2);
return (file_1, pass_value);
```

In the pseudo-code described above, key_1 is used as the result of XOR operation between the password provided by the user and the MAC address. The key_2 is a private key that will be present with authenticated users only, it can also be used for recovery purposes; secret_1 is a simple integer value that is kept private; file_1 is a byte stream and pass_value will be used for further encryption.

5 Results and Discussions

The expected outcome of this project is that it shall be used in industries where piracy causes huge amounts of loss. With more research into this field, copyright laws [14] will benefit a great deal in the future. It also has the possibility to become the norm for downloading and uploading of copyrighted content in the internet. Since, this algorithm produces a lossless copy of the original content, high quality multimedia content will consume reasonable amount of hard disk space while being secure (Fig. 4).

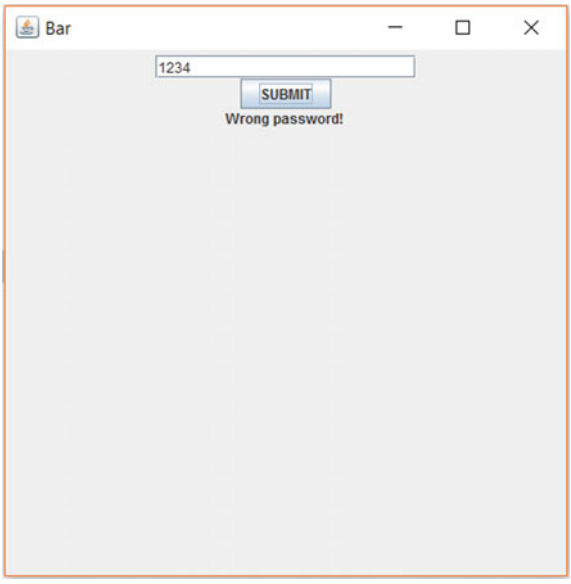

Fig. 4 The form for getting the password (in this case, a wrong password is entered)

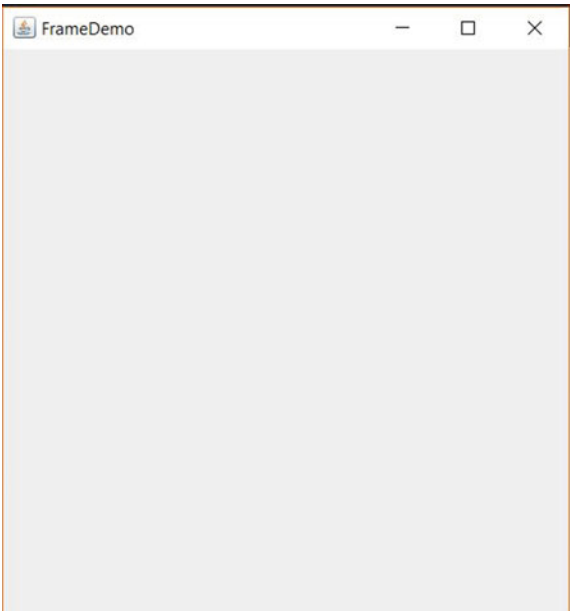

Fig. 5 This window will appear on successful entry of the password

When the Fig. 5 appears, a new directory named temp_folder will appear which will contain the decompressed files. The user will be unable to perform paste operation when this window is showing. Upon closing this window, the files will be deleted and will reappear again when the program is rerun.

Currently, we are able to achieve at best a 55–60% compression ratio for pdf file of 14 Megabytes. Some other tests include images, music and audio-video files, 60–65% in case of images, 80–90% in case of music and audio-video files. The time required for compressing depends upon the file size and the type of file. However, this technology seems to be working best with document files.

Below is a figure of a music file that has been compressed and encrypted successfully (Fig. 6).

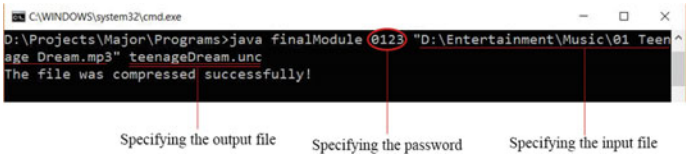

Fig. 6 A file successfully compressed

6 Limitations

- The key provided by the user has to be comprised of integer values only.
- The size of the software used for extracting the original files will remain the same at all times. This will take up more space during replay of the content.
- Better encryption and compression algorithms that work individually exist.
- It is not a final solution to provide fool proof security or data compression.

7 Future Scope

- The compression ratios can be better with different settings.
- The number of algorithms should be increased to provide better security.
- The number of rounds of encryption can be increased for better security.
- The algorithm can be optimized for its use in various regions.

8 Conclusions

This work is expected to help in industries where piracy auses huge amounts of loss. With more research into this field, copyright laws will benefit a great deal in the future. It also has the possibility to become the norm for downloading and uploading of copyrighted content in the internet. Since, this algorithm produces a lossless copy of the original content, high quality multimedia content will consume reasonable amount of hard disk space while being secure.

Acknowledgements Let us express our heartiest gratitude to respective authorities of Sikkim Manipal Institute of Technology, Sikkim, INDIA for providing resources used during the entire development process.

References

1. VidyaSagar M, Rose Victor JS (2013) Modified run length encoding scheme for high data compression rate. Int J Adv Res Comput Eng Technol (IJARCET) 2(12), December 2013
2. Zhang Y, Zhang LY, Zhou J, Liu L, Chen F, He X (2016) A review of compressive sensing in information security field. IEEE Access 4:2507–2519
3. Cappaert J, Kisserli N, Schellekens D, Preneel B (2006) Self-encrypting code to protect against analysis and tampering
4. Franceschini R, Mukherjee A (1996) Data Compression using encrypted text. In: Proceedings of Advanced Digital Libraries (ADL'96)

5. Mahoney M (2013) Data compression explained. Dell Inc.
6. Khelifi F, Brahimi T, Han J, Li X (2018) Secure and privacy-preserving data sharing in the cloud based on lossless image coding. Signal Process 148:91–101, ISSN 0165-1684
7. Alsheikh MA, Lin S, Niyato D, Tan HP (2016) Rate-distortion balanced data compression for wireless sensor networks. IEEE Sens J 16:5072–5083
8. Story A (2004) Intellectual property and computer software May 2004 intellectual property rights and sustainable development
9. Tyrväinen P, Järvi J, Luoma E (2004) Peer-to-peer marketing for content products combining digital rights management and multilevel marketing. In: Proceedings of IADIS International Conference on e-Commerce 2004 (EC2004) Lisbon, Portugal, 14–16 Dec 2004
10. Löytynoja M, Seppänen T, Cvejic N (2003) Experimental DRM architecture using watermarking and PKI. In: Proceeding of First International Mobile IPR Workshop Rights Management of Information Products on the Mobile Internet, Helsinki, Finland, 47–52
11. Jang KW, Park CK, Kim JJ, Jun MS (2006) A study on DRM system for on/off line key authentication. In: Proceedings of the 2006 International Conference on Security and Management, SAM 2006, Las Vegas, Nevada, USA, June 26–29
12. Stamp M (2003) Digital rights management: the technology behind the hype. J Electron Commer Res (August, 2003)
13. Wu CP, Jay Kuo CCJ (2005) Design of integrated multimedia compression and encryption systems. IEEE Trans Multimed 7(5), October 2005
14. De Hert P, Papakonstantinou V, Malgieri G, Beslay L, Sanchez I (2018) The right to data portability in the GDPR: towards user-centric interoperability of digital services. Comput Law Secur Rev, 34(2):193–203, ISSN 0267-3649