Terbit online pada laman web jurnal : **http://jurnal.iaii.or.id**

JURNAL RESTI

(Rekayasa Sistem dan Teknologi Informasi)

Vol. 2 No. 2 (2018) 486 - 491 ISSN Media Elektronik : 2580-0760

Implementasi Pengacakan *Bit* Nilai Warna Spasi Pada Steganografi Di Dokumen Teks OOXML

Tri Yatmo Noveary[a], Kevin Ekaputra[b], Halim Agung[c]

[a]Fakultas Teknologi dan Desain, Universitas Bunda Mulia, email: tri_yatmo_noveary@yahoo.co.id

[b]Fakultas Teknologi dan Desain, Universitas Bunda Mulia, email: kevin.ekaputra36@gmail.com

[c]Fakultas Teknologi dan Desain, Universitas Bunda Mulia, email: halim@bundamulia.ac.id

Abstract

Steganography exists due to the need to hide data in another data. The purpose of this research is to find another alternative in doing steganography via Office Open XML text document files. The research is done by library review, and application building. Design is done using flowcharts. Tests are done by comparing the result, and testing the endurance of the message via file format change, and content and name changing. From the result of the result, steganography done using space color code bit position randomization will not visually affect the document significantly. The message can endure conversions to file formats DOC, ODT, and PDF, 93% endure conversions to RTF file format, and cannot endure conversions to TXT file format, and can endure content change as long as there's the sequence of files, and the spaces left intact.

Keywords: Steganography, Office Open XML, space color code bits

Abstrak

Steganografi ada dikarenakan oleh adanya keperluan untuk menyembunyikan sebuah data di dalam data lain. Tujuan dari penelitian ini adalah untuk menemukan alternatif lain dalam melakukan steganografi melalui media berkas dokumen teks *Office Open* XML. Penelitian dilakukan dengan tinjauan pustaka, dan membuat komponen aplikasi. Perancangan dilakukan menggunakan *flowchart*. Pengujian dilakukan dengan cara membandingkan hasil, dan uji ketahanan melalui perubahan jenis berkas, dan perubahan konten, dan nama berkas. Berdasarkan hasil penelitian yang didapat, steganografi yang dilakukan dengan cara menambah pengacakan posisi *bit* kode warna spasi tidak memengaruhi dokumen secara signifikan secara visual. Pesan dapat tahan konversi ke jenis berkas DOC, ODT, dan PDF, 93% tahan konversi ke jenis berkas RTF, dan tidak tahan konversi ke jenis berkas TXT, dan dapat tahan perubahan konten, asalkan urutan berkas tetap ada, dan spasi tidak diganggu.

Kata kunci: Steganografi, Office Open XML, *Bit* Kode Warna Spasi.

1. Pendahuluan

Data di dunia ini berdasarkan visibilitasnya ada yang bersifat publik, dengan kata lain dapat dan boleh dilihat oleh siapa saja, dan ada yang bersifat privat, dengan kata lain tidak dapat dan tidak boleh dilihat oleh sembarang orang. Contoh data yang bersifat publik adalah entri pada sebuah kamus, iklan, dan *source code* aplikasi *open source*. Sedangkan, contoh data yang bersifat privat adalah kata sandi, pesan rahasia, dan data sensitif secara umum. Data publik dapat dan boleh dilihat oleh umum, dan data privat tidak dapat dan tidak boleh dilihat oleh umum. Dengan kata lain, data privat harus disembunyikan.

Ada beberapa cara umum untuk menyembunyikan data. Salah satu cara adalah dengan mencegah akses ke tempat data tersebut berada. Selain itu, cara menyembunyikan data adalah dengan menerjemahkannya ke dalam bahasa kode, dengan kata lain, enkripsi. Ada juga cara menyembunyikan data dengan cara menyembunyikan data tersebut ke dalam data lain, dengan kata lain, steganografi.

Steganografi ini sendiri dapat dikatakan merupakan seni menyembunyikan data (atau dalam kasus dunia nyata, pesan teks, atau berkas) ke dalam data lain (atau dalam kasus dunia nyata, satu berkas atau lebih). Berkas-berkas yang digunakan dalam steganografi bermacam-macam. Ada yang menggunakan gambar, suara, *video*, *executable*, dan teks, baik yang berformat maupun yang tidak berformat.

Steganografi dengan media teks berformat dapat menggunakan beberapa metode. Ada yang menggunakan metode yang sama dengan metode

Diterima Redaksi : 11-03-2018 | Selesai Revisi : 17-05-2018 | Diterbitkan Online : 07-06-2018

steganografi berbasis teks tak berformat, ada juga yang menggunakan format-format yang ada. Ada yang menggunakan warna, dan teks tak tampak.

Sebelum dilakukan penelitian ini, Dwi Fegiannata dalam skripsi yang berjudul "Implementasi Pengacakan *Pixel* Pada Metode Steganografi LSB dengan Java" telah meneliti tentang penerapan pengacakan *pixel* untuk LSB pada gambar [4]. Selain itu, Dorisman Putra [11] dalam skripsi yang berjudul "Teknik Steganografi pada *Rich Text Format File* Dengan Memanfaatkan Atribut Warna Teks" telah meneliti tentang penerapan atribut warna pada berkas RTF untuk menyisipkan pesan. Lalu, Md. Khairullah dalam penelitian yang berjudul "*A Novel Text Steganography System Using Font Color of the Invisible Characters in Microsoft Word Documents*" telah meneliti tentang penggunaan warna karakter tak terlihat untuk menyisipkan pesan. Penelitian-penelitian tersebut adalah yang menjadi dasar untuk penelitian ini.

Dalam penelitian ini, peneliti mengimplementasikan bagian metode pengacakan SHA1 dari *library* SecureRandom Java edisi 8 yang diekspor ke bahasa Visual Basic .NET untuk mengacak posisi penempatan pesan pada warna spasi.

2. Tinjauan Pustaka

Office Open XML adalah jenis XML yang digunakan untuk mendefinisikan dokumen Microsoft Office seperti yang telah dijelaskan di dalam standar ECMA-376-1:2016 (Ecma International, 2016). *Open Office* XML mencakup dokumen teks Microsoft Office (nama resminya WordprocessingML), dokumen *spreadsheet* Microsoft Office (nama resminya SpreadsheetML), dokumen presentasi Microsoft Office (nama resminya PresentationML), gambar yang digunakan di dalam dokumen Microsoft Office (nama resminya DrawingML), dan pengemasan dokumen-dokumen tersebut. Office Open XML menggunakan teknologi ZIP untuk mengemas berkas-berkas XML dan berkas-berkas lain yang terlibat dalam dokumen [1].

Beberapa hal yang relatif umum ada di dalam sebuah bagian utama dokumen *Office Open* XML adalah:

Paragraf mendefinisikan pembagian konten dalam bentuk baris baru. Paragraf memiliki konten dan sifat. Konten biasanya berbentuk *run*. Beberapa jenis konten yang terdapat di dalam paragraf di antaranya sifat *run*, teks, *line break*, *field*, tab, nomor halaman, konten DrawingML, gambar, konten XML lain, *carriage return*, dan simbol. Beberapa sifat *run* yang dapat diatur adalah cetak tebal, cetak miring, arah teks, bingkai teks, kapitalisasi, warna teks, garis tengah, efek timbul, efek ukir, efek kontur, ukuran teks, penggunaan garis bawah, dan skala ukuran teks. Warna teks yang digunakan dapat menggunakan warna tema yang ada, menggunakan warna sendiri, didefinisikan di dalam berkas dalam bentuk nilai heksadesimal. Dua karakter pertama adalah nilai warna merah dalam heksadesimal, dua karakter kedua adalah nilai warna hijau dalam heksadesimal, dan dua karakter terakhir adalah nilai warna biru dalam heksadesimal.

Tabel dapat didefinisikan sebagai kumpulan paragraf yang diatur di dalam baris-baris dan kolom-kolom.

Steganografi adalah tindakan komunikasi secara tersembunyi [2]. Kata steganografi berasal dari kata bahasa Yunani, yaitu *steganos*, yang berarti tutup, dan kata bahasa Latin *graphia*, yang berarti tulisan. Dengan kata lain, steganografi berarti seni komunikasi tersembunyi.

Jenis-jenis steganografi berdasarkan media penyimpanannya di antaranya Steganografi pada gambar dapat dilakukan dengan cara memanfaatkan LSB, yang dapat menerapkan pengacakan posisi *pixel* [4]. Selain itu, terdapat juga metode tersendiri untuk steganografi ke jenis gambar JPEG [5], dan SVG [6]. Steganografi pada media *audio*/suara/bunyi dapat dilakukan dengan cara memanfaatkan LSB [7]. Steganografi pada media *video* dapat dilakukan dengan cara memanfaatkan LSB. Selain LSB ini, ada juga yang menerapkan kode *Hamming* (7,4) untuk menyisipkan pesan ke berkas *video* [8]. Steganografi pada media teks dapat dilakukan dengan berbagai cara bergantung pada jenis format teks. Beberapa cara yang dapat digunakan adalah menggunakan RSID, kompresi ZIP yang berbeda, menggunakan gambar berdimensi 0, menggunakan *macro* [9], dan menggunakan warna teks tak terlihat, seperti spasi [10].

Pseudorandom number generator disebut juga *deterministic random bit generator*. Dalam konteks pembuatan angka acak, angka acak adalah sebuah nilai di dalam sebuah himpunan yang memiliki kemungkinan sama untuk dipilih dari populasi kemungkinan, sehingga nilai tersebut tidak dapat diprediksi. Sementara itu, *pseudo-random* berarti proses atau data yang hasilnya deterministik, atau pasti, tetapi tetap secara efektif (efektif dalam konteks ini berarti masih di dalam batasan kekuatan kriptografis yang diinginkan) acak, selama kejadian-kejadian di dalam proses tersebut tidak diawasi [3].

3. Metodologi Penelitian

Metodologi penelitian yang digunakan dalam penelitian ini adalah melalui *library research*, dan *laboratory research*. Penelitian dimulai dari *library research*, yaitu mencari bahan dan kepustakaan mengenai apa saja yang berhubungan dengan penelitian ini. Kemudian, dilakukan *laboratory research*, yaitu aplikasi mulai dirancang, dibangun, dan diuji, apakah berjalan seperti yang diharapkan. Implementasi dilakukan dengan menggunakan bahasa Visual Basic .NET dengan tambahan *library* OpenXML SDK 2.5. Kemudian, pengujian dilakukan dengan cara membandingkan

secara visual dokumen asal, dan dokumen hasil, dan juga percobaan konversi ke *format* PDF, ODT, DOC, TXT, dan RTF, dan dikembalikan ke *format* Office Open XML, perubahan nama, dan perubahan isi. Perangkat lunak yang digunakan dalam konversi ke *format* PDF, ODT, DOC, TXT, dan RTF, dan sebaliknya adalah LibreOffice 5.1.6.2. Selain itu, dilakukan juga pencatatan perbedaan besar sebagai referensi.

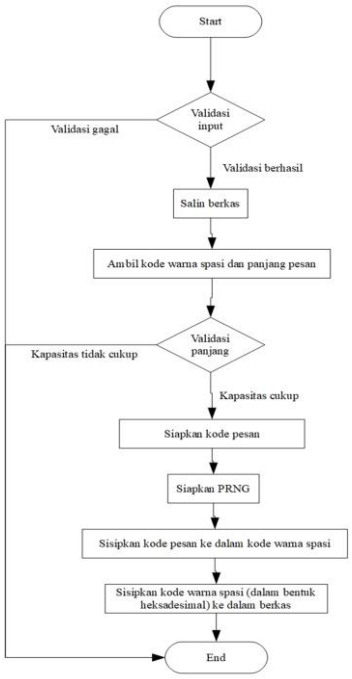
4. Hasil dan Pembahasan

Sistem yang akan dibuat adalah sistem aplikasi penyisipan pesan ke dalam media berkas dokumen *Office Open* XML. Metode penyisipan yang akan digunakan adalah LSB pada warna spasi dengan pengacakan posisi *bit* warna. *Bit* warna diambil dari kode warna spasi-spasi yang ada di dokumen. Agar urutan angka acak dapat diingat kembali, digunakan PRNG untuk mengacak posisi penyisipan. Untuk memudahkan pengguna mengingat kunci, kata sandi dijadikan kunci untuk PRNG. Berkas dokumen *Office Open* XML yang digunakan dapat lebih dari satu. Oleh karena itu, semua kode warna spasi pada tiap berkas dokumen disatukan ke dalam satu kode warna spasi.

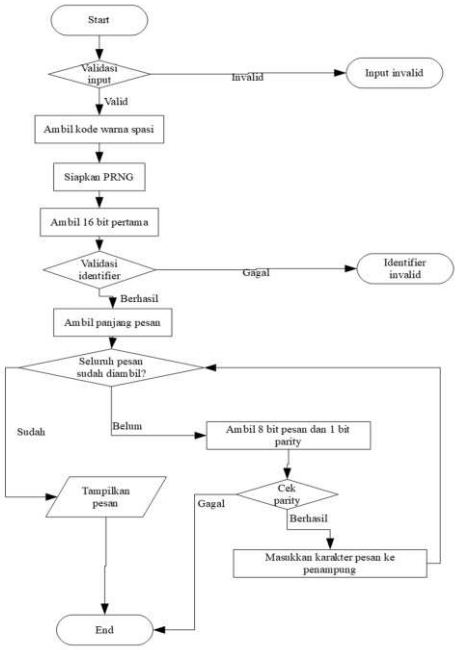
Flowchart untuk penyisipan pesan ke berkas dokumen teks *Office Open* XML terdapat pada Gambar 1. *Flowchart* untuk pengambilan pesan dari berkas dokumen teks Office Open XML terdapat pada Gambar 2. *Flowchart* untuk persiapan PRNG terdapat pada Gambar 3.

Berikut ini adalah penjelasan dari Gambar 1. Pertama, masukan berupa pesan, dan berkas divalidasi. Jika gagal, maka operasi berakhir. Kegagalan ini diakibatkan oleh pesan yang kosong atau tidak ada berkas yang menjadi tempat penyisipan. Jika berhasil, maka berkas disalin, lalu ambil kode warna spasi dari berkas dan juga siapkan panjang pesan. Kemudian, kapasitas berkas-berkas dicek. Jika mencukupi, maka kode pesan disiapkan, lalu PRNG disiapkan dengan kata sandi (ada masukan kata sandi), atau tanpa kata sandi (tidak ada masukan kata sandi). Lalu, ubah *bit-bit* yang ditunjuk oleh PRNG menjadi *bit-bit* yang ada di kode pesan. Cara PRNG menunjuk adalah dengan menunjuk nilai di antara 0 sampai panjang *string* kode warna spasi. Jika posisi sudah digunakan, maka proses diulang sampai didapat posisi yang belum digunakan. Akhirnya, sisipkan kode warna spasi kembali ke dalam berkas-berkas.

Berikut ini adalah penjelasan dari Gambar 2. Pertama, masukan berupa berkas divalidasi. Jika gagal, maka operasi berakhir. Jika berhasil, maka kode warna spasi berkas-berkas diambil, lalu PRNG disiapkan dengan *password*, atau tanpa *password.*



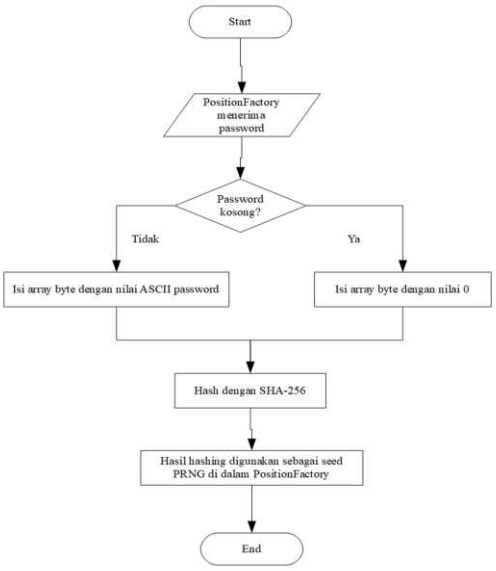
Gambar 1. *Flowchart* penyisipan pesan.



Gambar 2. *Flowchart* pengambilan pesan.

Kemudian 16 *bit* pertama diambil seperti ditunjuk oleh PRNG. *Bit-bit* ini adalah *identifier* yang dimasukkan bersama dengan pesan dan panjang pesan. Kemudian *identifier* divalidasi (Apakah *bit-bit* tersebut berisi nilai "0100010001000110"). Jika *identifier* tidak valid, maka operasi dihentikan. Jika valid, maka panjang

pesan diambil (32 *bit* selanjutnya). Kemudian, ambil pesan sampai seluruh pesan sudah diambil, yaitu ketika panjang pesan yang telah diambil sama dengan panjang pesan yang diambil dari 32 *bit* setelah 16 *bit identifier*. Sementara mengambil pesan, ada juga pengecekan *parity*. Jika pengecekan berhasil, maka pesan dimasukkan ke dalam penampung pesan. Jika gagal, maka operasi dihentikan. Setelah seluruh pesan sudah diambil, maka pesan ditampilkan.



Gambar 3. *Flowchart* persiapan PRNG.

Berikut ini adalah penjelasan dari Gambar 3. Pertama, kata sandi yang telah disediakan diambil. Jika kata sandi tidak kosong, maka isi *array byte* dengan nilai ASCII *password*. Jika tidak, maka isi *array byte* dengan nilai 0. Setelah itu, *array byte* dilakukan *hashing* dengan algoritma SHA-256, kemudian nilai *hash* tersebut digunakan sebagai *seed* untuk PRNG.

Dari 30 percobaan, tidak terdapat perubahan yang terdeteksi mata, dan seluruh pesan dapat diambil dengan sempurna untuk yang tidak dikonversi ke bentuk-bentuk lain seperti ODT, PDF, RTF, TXT, atau DOC (dengan kata lain, masih murni DOCX). Ketidakterlihatan ini disebabkan oleh karena sifat spasi yang dari warna hitam (kode heksadesimalnya 000000, dengan nilai *channel* merah 0, nilai *channel* hijau 0, dan nilai *channel* biru 0) hingga putih (kode heksadesimalnya ffffff, dengan nilai *channel* merah 255, nilai *channel* hijau 255, dan nilai *channel* biru 255), tidak memiliki perbedaan visual. Ini juga berarti bahwa setiap spasi memiliki kapasitas 24 *bit* untuk menyimpan data/pesan.

Dari 30 percobaan, nama berkas yang diubah tidak akan merusak pesan, asalkan urutan pengambilan pesan sama, dan isi berkas yang diubah tidak akan merusak pesan asalkan tidak ada perubahan spasi, baik jumlah maupun sifat spasi.

Dari 30 percobaan, hasil uji ketahanan konversi dapat dilihat pada Tabel 1. Uji ketahanan konversi dilakukan dengan cara mengubah berkas berisi pesan ke dalam jenis tertentu, lalu dikembalikan ke jenis sebelumnya.

Tabel 1. Tabel hasil uji ketahanan.

Uji	ODT	PDF	RTF	TXT	DOC
1	Ya	Ya	Ya	Tidak	Ya
2	Ya	Ya	Ya	Tidak	Ya
3	Ya	Ya	Ya	Tidak	Ya
4	Ya	Ya	Ya	Tidak	Ya
5	Ya	Ya	Tidak	Tidak	Ya
6	Ya	Ya	Ya	Tidak	Ya
7	Ya	Ya	Ya	Tidak	Ya
8	Ya	Ya	Ya	Tidak	Ya
9	Ya	Ya	Tidak	Tidak	Ya
10	Ya	Ya	Ya	Tidak	Ya
11	Ya	Ya	Ya	Tidak	Ya
12	Ya	Ya	Ya	Tidak	Ya
13	Ya	Ya	Ya	Tidak	Ya
14	Ya	Ya	Ya	Tidak	Ya
15	Ya	Ya	Ya	Tidak	Ya
16	Ya	Ya	Ya	Tidak	Ya
17	Ya	Ya	Ya	Tidak	Ya
18	Ya	Ya	Ya	Tidak	Ya
19	Ya	Ya	Ya	Tidak	Ya
20	Ya	Ya	Ya	Tidak	Ya
21	Ya	Ya	Ya	Tidak	Ya
22	Ya	Ya	Ya	Tidak	Ya
23	Ya	Ya	Ya	Tidak	Ya
24	Ya	Ya	Ya	Tidak	Ya
25	Ya	Ya	Ya	Tidak	Ya
26	Ya	Ya	Ya	Tidak	Ya
27	Ya	Ya	Ya	Tidak	Ya
28	Ya	Ya	Ya	Tidak	Ya
29	Ya	Ya	Ya	Tidak	Ya
30	Ya	Ya	Ya	Tidak	Ya

Tabel 1 berisi hasil uji ketahanan konversi. Pesan sepenuhnya tahan terhadap uji konversi ke jenis berkas ODT, PDF, dan DOC. Ini disebabkan oleh karena seluruh warna teks dapat dijaga. Lalu, pesan tidak sepenuhnya tahan terhadap uji konversi ke jenis berkas RTF, yaitu 93%. Ini disebabkan oleh karena hanya sebagian warna teks yang dapat dijaga. Kemudian, pesan sama sekali tidak tahan konversi ke jenis berkas TXT. Ini disebabkan oleh karena tidak ada bagian sifat

yang dapat dibawa ke TXT, sehingga jika dikembalikan, tidak akan ada sisa pengaturan sifat pada teks.

Dari 30 percobaan, perubahan besar dari masing-masing berkas telah dirangkum dan dapat dilihat pada Tabel 2.

Tabel 2. Tabel Selisih besar.

Uji	Besar (byte)	Besar setelah (byte)	Selisih besar (byte)	Selisih besar (%)
1	4659	6389	1730	37,13%
	4128	5755	1627	39,41%
2	1245614	1246722	1108	0,09%
3	6574	7235	661	10,05%
4	9534	12599	3065	32,15%
5	9534	10502	968	10,15%
	4659	4938	279	5,99%
	6574	6924	350	5,32%
6	9534	10494	960	10,07%
	1245614	1246236	622	0,05%
7	4659	5199	540	11,59%
8	4128	4710	582	14,10%
	6574	9731	3157	48,02%
	9534	17368	7834	82,17%
9	1245614	1252352	6738	0,54%
	4659	7324	2665	57,20%
	4128	6631	2503	60,63%
	6574	6593	19	0,29%
	9534	9502	-32	-0,34%
10	1245614	1245663	49	0,00%
	4659	4680	21	0,45%
	4128	4161	33	0,80%
11	6574	7343	769	11,70%
12	9534	10556	1022	10,72%
13	1245614	1247070	1456	0,12%
14	4659	4980	321	6,89%
15	4128	4261	133	3,22%
16	6574	7159	585	8,90%
17	9534	13508	3974	41,68%
18	1245614	1246641	1027	0,08%
19	4659	5714	1055	22,64%
20	4128	4867	739	17,90%
21	6574	7018	444	6,75%
22	9534	10849	1315	13,79%
23	1245614	1246156	542	0,04%
24	4659	4888	229	4,92%
25	4128	4628	500	12,11%
26	6574	7929	1355	20,61%
27	9534	11382	1848	19,38%
28	1245614	1247395	1781	0,14%
29	4659	5918	1259	27,02%
30	4128	6307	2179	52,79%

Pada uji 10, berkas kedua, terjadi penyusutan besar. Salah satu kemungkinan penyebab kejadian tersebut adalah karena selain untuk mengisi berkas dengan kode warna yang berisi pesan rahasia, aplikasi juga memiliki efek samping berupa menyatukan *run-run* dengan sifat sama.

Dari uji-uji tersebut, dapat ditarik kesimpulan bahwa berkas dokumen *Office Open* XML yang berisi pesan dapat diubah menjadi jenis-jenis berkas ODT, DOC, dan PDF tanpa merusak pesan. Akan tetapi, berkas hanya 93% tahan terhadap konversi ke RTF, dan sama sekali tidak tahan terhadap konversi ke TXT. Secara keseluruhan, penerapan pengacakan posisi *bit* kode warna spasi pada steganografi Selain itu, perubahan nama dapat dilakukan, asalkan tidak merubah urutan. Lalu, boleh dilakukan perubahan isi, asalkan spasi tidak diganggu, baik dengan penambahan, maupun pengurangan. Lalu, tidak ada perbedaan yang kasat mata di antara berkas awal, dan berkas yang disisipi pesan.

5. Kesimpulan

Setelah dilakukan penelitian terhadap penerapan pengacak posisi *bit* nilai warna spasi pada steganografi pada berkas dokumen teks *Office Open* XML, maka didapatkan hasil kesimpulan dan beberapa saran.

Implementasi pengacakan *bit* nilai warna spasi pada steganografi di berkas dokumen teks *Office Open* XML dapat dilakukan dengan memanfaatkan PRNG yang menggunakan *seed* yang berasal dari kata sandi yang disediakan pengguna dengan persentase keberhasilan uji ketahanan DOC, ODT, PDF, TXT, dan RTF adalah 78,67%.

Beberapa saran untuk penelitian di masa mendatang adalah menerapkan gambar, atau gabungan gambar dan warna teks pada dokumen teks Office Open XML untuk menyembunyikan pesan berkas, dan menerapkan kode *Hamming* sebagai pencegah galat di dalam pesan yang akan disisipkan.

6. Daftar Rujukan

[1] *ECMA-376-1:2016 Office Open XML File Formats – Fundamentals and Markup Language Reference*, ECMA-376-1:2016, 2016.

[2] I.J. Cox, M.L. Miller, J.A. Bloom, J. Fridrich, dan T. Kalker, *Digital Watermarking and Steganography,* 2nd ed. Burlington, Morgan Kaufmann, 2007.

[3] E. Barker, dan J. Kelsey, *Recommendation for Random Number Generation Using Deterministic Random Bit Generators,* 2015. [Online]. Available: https://www.nist.gov. Accessed on 21 September 2017

[4] D. Fegiannata, "Implementasi Pengacakan Pixel Pada Metode Steganografi LSB dengan Java," Skripsi S. Kom., Fakultas Teknologi dan Desain, Universitas Bunda Mulia, Jakarta, Indonesia, 2014.
[5] A. Westfeld, "F5 – A Steganographic Algorithm," di *4th International Information Hiding Workshop*, Pittsburgh, 2001, pp 289-301.
[6] B. Madoš, J. Hurtuk, M. Čopjak, P. Hamaš, dan M. Ennert, "Steganographic Algorithm for Information Hiding Using Scalable Vector Images," *Acta Electrotechnica et Informatica*, vol.14, no. 4, pp. 42-45, Desember, 2014. doi: 10.15546/aeei-2014-0040, [Online], Available: http://www.aei.tuke.sk/.
[7] K. Gopalan, "Audio Steganography Using Bit Modification," di *IEEE International Conference on Acoustics, Speech and Signal Processing*, Juli, 2003. doi: 10.1109/ICASSP.2003.1202390, [Online], Available: https://www.researchgate.net/.
[8] R.J. Mstafa, K.M. Elleithy. "A Highly Secure Video Steganography using Hamming Code (7, 4)," di *Proceedings of Systems, Applications and Technology Conference (LISAT)*, Mei, 2014, doi: 10.1109/LISAT.2014.6845191, [Online], Available: https://scholarworks.bridgeport.edu/.
[9] A. Castiglione, B. D'Alessio, A. De Santis, F. Palmieri, "New Steganographic Techniques for the OOXML File Format," di *Availability, Reliability and Security for Business, Enterprise and Health Information Systems*, Agustus, 2011, pp 344-358, doi: 10.1007/978-3-642-23300-5_27, [Online], Available: https://link.springer.com/.
[10] M. Khairullah, "A Novel Text Steganography System Using Font Color of the Invisible Characters in Microsoft Word Documents," di *Second International Conference on Computer and Electrical Engineering, 2009*, Desember, 2009, doi: 10.1109/ICCEE.2009.127, [Online]. Available: http://ieeexplore.ieee.org/.
[11] D. Putra, "Teknik Steganografi pada *Rich Text Format File* Dengan Memanfaatkan Atribut Warna Teks," Skripsi S. Kom., Fakultas Matematika dan Ilmu Pengetahuan Alam, Institut Pertanian Bogor, Bogor, 2010.
[12] Čopjak, Marek, dkk., (2014), Steganographic Algorithm for Information Hiding Using Scalable Vector Images, Acta Electrotechnica et Informatica, Vol 14, issue 4, pp 42-45. http://www.aei.tuke.sk. Accessed on 5 November 2017.
[13] Mishra, Nishchol, Sharma, Sanjeev, dan Yadav, Pooja, (2013), A secure video steganography with encryption based on LSB technique, 2013 IEEE International Conference on Computational Intelligence and Computing Research, Vol 2, pp 1-5. https://ieeexplore.ieee.org/. Accessed on 23 November 2017.