Watermarking pada Cross Reference(XRef) Portable Document Format(PDF) dengan Enkripsi RC4

Rohmat Gunawan[1], Rinaldi Munir[2]

Program Studi Magister Informatika, Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung
Jalan Ganesha No. 10, Bandung, Jawa Barat, 40132
[1]rohmatgunawan@unsil.ac.id, [2]rinaldi-m@stei.itb.ac.id

Abstrak—***Portable Document Format (PDF)*** **merupakan salah satu jenis berkas yang paling banyak disebarkan melalui internet. Dengan semakin meningkatnya penggunaan format *PDF*, perlindungan hak cipta terhadap *PDF* merupakan salah satu masalah yang perlu diperhatikan terutama untuk membuktikan kepemilikan dokumen. *Watermarking* merupakan salah satu metode yang dapat digunakan untuk melindungi hak cipta pada dokumen digital. Struktur internal berkas PDF terdiri dari : *header, body, xref* dan *trailer*. *Watermarking* pada berkas PDF umumnya disisipkan di bagian *body*, dan mudah dideteksi sehingga mudah dihapus oleh *tools* tertentu. Penyisipan *watermark* dengan cara menimpa *character ASCII 20* di bagian *xref* dan penyisipan *watermark* di bagian *xref* baru (hasil *incremental update*) merupakan alternatif yang telah berhasil dikembangkan dalam penelitian. Tetapi, karena *xref* mempunyai ukuran terbatas, sehingga ketika *watermark* melebihi kapasitas *xref*, sebagian *watermark* tidak dapat disisipkan. Sedangkan penyisipan *watermark* di bagian *xref* baru menyebabkan ukuran berkas setelah proses penyisipan selalu bertambah. Oleh karena itu, diusulkan untuk melakukan penyisipan *watermark* di bagian *generation number xref*. Sebelum disisipkan *watermark* dienkripsi menggunakan algoritma RC4 agar tidak mudah dipersepsi. Hasil eksperimen menunjukkan bahwa, penyisipan *watermark* di bagian *generatiom number xref* tidak menambah ukuran berkas selama kapasistas *xref* yang ada mampu menampung seluruh *watermark*. Ketika *watermark* melebihi kapasistas *xref*, maka dilakukan *additional xref* sehingga seluruh *watermark* dapat disisipkan.**

***Kata Kunci : Watermarking, PDF, XRef*, Enkripsi RC4.**

I. PENDAHULUAN

Internet merupakan salah satu media saat ini yang digunakan untuk penyebaran dokumen digital. *Portable Document Format (PDF)* merupakan salah satu format dokumen yang paling banyak disebarkan melalui internet. Menurut laporan survei, penggunaan kertas untuk penyimpanan jangka panjang akan turun menjadi 77% selama lima tahun berikutnya, sedangkan PDF naik sampai 93% [1]. Dengan semakin meningkatnya penggunaan format *PDF* ini, perlindungan hak cipta terhadap *PDF* merupakan salah satu masalah yang perlu diperhatikan terutama untuk membuktikan kepemilikan dokumen tersebut.

Watermarking merupakan salah satu metode yang dapat digunakan untuk melindungi hak cipta pada dokumen digital [5]. *Watermarking* dapat diterapkan pada dokumen multimedia digital, seperti : *image, audio, video* dan *text*. Berbagai pendekatan *(approach) text watermarking* seperti : *image based approach* [3], [4] *syntactic approach*, *semantic approach* dan *structural approach* diusulkan dalam berbagai penelitian untuk diterapkan pada *PDF watermarking*. Pada penelitian ini dikembangkan metode *watermarking* pada berkas *PDF* dengan pendekatan berbasis struktur (*structural approach)*. Pendekatan berbasis struktur yang dilakukan , bukan pada struktur kalimat yang terdapat pada berkas *PDF* tetapi pada struktur internal berkas *PDF*. Struktur internal berkas *PDF* terdiri dari 4 bagian : *header, body, xref* dan *trailer* seperti terlihat pada Gambar 1.

Gambar 1. Struktur internal berkas *PDF*

Watermark pada berkas PDF umumnya ditanamkan di bagian *body*. *Watermark* yang ditanamkan di bagian *body* mudah dideteksi dan dapat dihapus menggunakan *tools* tertentu seperti **PDF Watermark Remover**.

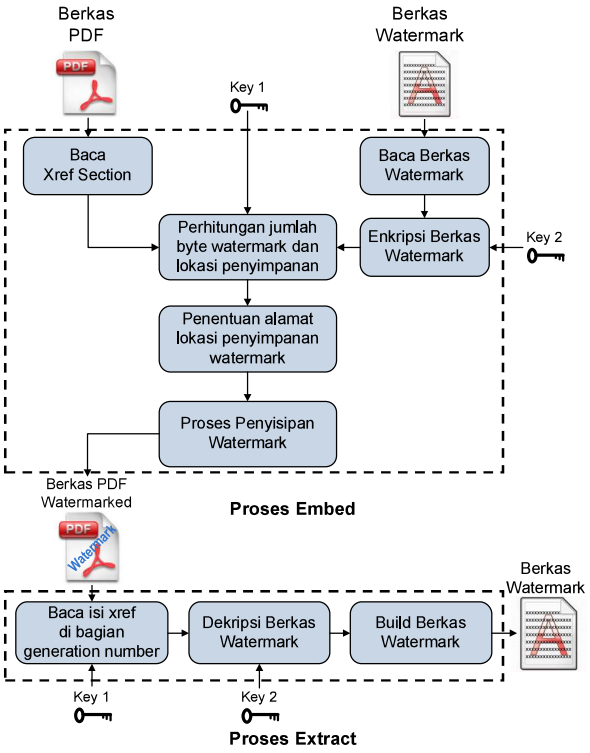
II. ULASAN PENELITIAN TERKAIT

Penyisipan *watermark* di bagian *xref* *PDF* sebelumnya telah dikembangkan dalam penelitian [2] [9]. Pada penelitian [2], penyisipan *watermark* dilakukan dengan cara menimpa *3 byte character 20 ASCII* atau karakter spasi dalam setiap baris *xref*. Sehingga *watermark* maksimum yang dapat disisipkan adalah 3 *byte* dikali banyaknya baris *xref*. Penelitiannya telah berhasil menyisipkan *watermark* di bagian *xref*, dan ukuran berkas *PDF* setelah proses penyisipan tidak bertambah. Tetapi metode yang diusulkan dalam penelitian tersebut belum dapat menangani penyisipan data *watermark* yang melebihi kapasitas *xref*

Penelitian lainnya [9], mencoba menyisipkan *watermark* di bagian *xref* baru hasil *incremental update*. Percobaan pada penelitian tersebut telah berhasil menyisipkan *watermark* di bagian *xref* baru. Ukuran berkas *PDF* setelah proses penyisipan *watermark* bertambah sekitar 1,5 %. Oleh karena itu dicoba alternatif lain penyisipan *watermark*, agar menjadi solusi bagi permasalahan pada kedua penelitian tersebut. Berdasarkan studi pustaka dan percobaan maka diusulkan untuk menyisipkan data *watermark* di bagian *generation number xref*.

III. USULAN METODE

Secara umum metode yang diusulkan pada penelitian ini terdiri dari dua bagian utama yaitu proses *embed* dan proses *extract* seperti terlihat pada Gambar.2



Gambar 2. Metode *PDF watermarking* yang diusulkan

A. *Proses Embed*

Pada penelitian ini proses penyisipan *watermark* pada berkas *PDF* terdiri dari 6 bagian dengan tahapan sebagai berikut :

1. Baca *xref section*

   Membaca *Xref Section* atau bagian *xref* pada berkas *PDF* merupakan langkah awal yang harus dilakukan. Tahapan ini dilakukan untuk mengetahui jumlah entri *xref*. Adapun proses yang dilakukan pada tahapan ini sebagai berikut :

   a. Mendapatkan ukuran entri *xref* dengan bantuan *keyword* **`/Size`** pada *trailer*. Ukuran entri *xref* disimpan pada *trailer* seteleh *keyword* **`/Size.`**

   Contoh terdapat *trailer* sebagai berikut :

```
trailer
<< /Size  22
/Root  2 0 R
/Info 1 0 R
/ID [ < 81b14aafa313db63dbd6f981e49f94f4 >
< 81b14aafa313db63dbd6f981e49f94f4 >
]
>>
startxref
18799
%%EOF
```

   Dari contoh tersebut dapat diketahui bahwa jumlah entri *xref* = 22 baris

   b. Menyimpan nilai ukuran entri *xref* ke dalam suatu variabel

   c. Mendapatkan jumlah *byte generation number* dengan perhitungan sebagai berikut :
   Jumlah *byte generation number* = Jumlah entri *xref* dikali lima (5).

2. Baca berkas *watermark*
   Tahapan berikutnya yaitu, membaca berkas yang akan disisipkan *(watermark file)*. Pembacaan berkas *watermark* dilakukan *per byte*.

3. Enkripsi berkas *watermark* dengan input *key* 2
   Tahapan selanjutnya yaitu, enkripsi berkas *watermark* dengan algoritma RC4 dengan input *key* 2. Proses enkripsi ini dilakukan agar berkas *watermark* tidak mudah dipersepsi. *Key* 2 adalah kunci yang digunakan sebagai input untuk melakukan proses enkripsi dengan algoritma RC4.

4. Perhitungan jumlah *byte watermark* dan lokasi penyimpanan dengan input *key* 1

   Sebelum *watermark* disisipkan maka dilakukan perhitungan ukuran atau jumlah *byte watermark*. Hal ini dilakukan untuk mengetahui jumlah *byte* yang akan disisipkan, dan dibandingkan dengan kapasitas *xref* yang tersedian, sehingga seluruh *watermark* dapat disisipkan di lokasi *xref*. Tahapan yang dilakukan pada proses ini sebagai berikut :

   a. Menghitung banyaknya kelompok *byte* dari berkas *watermark*. Misal : ukuran berkas *watermark* = 100 *byte*,

maka banyaknya kelompok *byte* tersebut adalah 100/5 = 20 kelompok. Dari hasil perhitungan tersebut dapat disimpulkan bahwa, minimal harus tersedia 20 entri atau baris lokasi penyimpanan pada *xref* untuk menampung seluruh data *watermark*.

b. Menghitung ukuran lokasi penyimpanan *watermark*. Ukuran atau kapasistas penyimpanan *watermar*k ditentukan oleh banyaknya atau jumlah entri *xref* .

c. Jika kapasitas *xref* tidak cukup untuk menampung seluruh *watermark*, maka akan dilakukan pertambahan entri *xref (additional xref)*, tetapi jika kapasitas *xref* mencukupi maka tidak dilakukan penambahan entri.

Misal :Jumlah kelompok *byte watermark* = 20. Jumlah entri *xref* = 15. Kapasistas daya tampung *watermark* 15 – 1 = 14. Karena jumlah baris *xref* kurang dari jumlah kelompok *byte watermark*, maka dilakukan pertambahan entri *xref* sebanyak 20 - 14= 6 baris.
Kapasitas daya tampung *watermark* dikurangi satu karena entri *xref* baris terakhir tidak digunakan untuk menyimpan data *watermark*, tetapi digunakan untuk menyimpan data jumlah kelompok *byte watermark*.

d. Selanjutnya, perhitungan lokasi penyimpanan *watermark* dilakukan dengan cara mengacak (random) urutan baris pada *xref*. Pada tahap ini *key* 1 digunakan untuk membangkitkan bilangan acak.

5. Penentuan alamat lokasi penyimpanan *watermark*
Data *watermark* akan disimpan di setiap entri *xref* secara acak. Pengacakan lokasi penyimpanan ini dilakukan agar urutan data *watermark* tidak mudah ditebak.

6. Proses penyisipan *watermark*.

Proses penyisipan *watermark* dilakukan dengan menimpa setiap *byte xref* khususnya di bagian *generation number*. Tahapan pada proses penyisipan ini sebagai berikut :

a. Baca data *watermark* yang telah dikelompokkan dan disimpan dalam *array*
b. Tulis setiap *byte generation number* dengan dengan data *watermark* sesuai dengan alamat yang telah disiapkan sebelumnya.

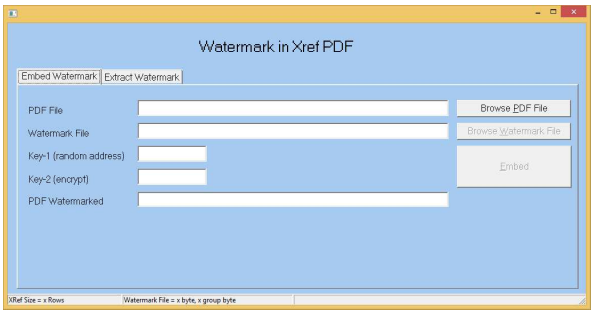
B. *Proses Extract*

Tahapan dalam proses ekstraksi ini terdiri dari :

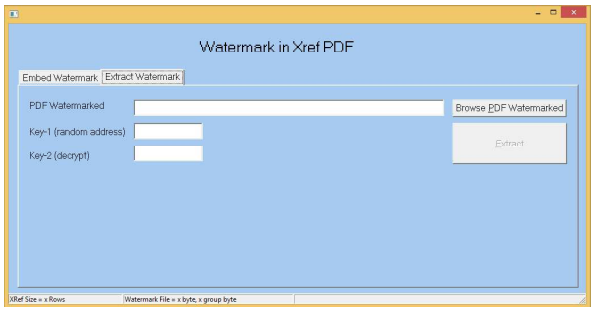
1. Baca *xref* di bagian *generation number* dengan *input key 1*
   a. Baca *key* untuk membangkitkan bilangan acak
   b. Dapatkan alamat lokasi penyisipan *watermark* berdasarkan bilangan acak yang dibangkitkan
   c. Baca entri *xref* di bagian *generation number*.
2. Dekripsi berkas *watermark* menggunakan algoritma RC 4 dengan input *key 2*.
3. *Build file watermark*
Tahap ini merupakan bagian terakhir dari proses ekstraksi. Setelah setiap *byte watermark* dibaca, selanjutnya disusun secara berurutan *(sequential)* sehingga tergabung dan akan mebentuk satu berkas *watermark* utuh.

IV. IMPLEMENTASI DAN PENGUJIAN

Metode yang diusulkan selanjutnya diimplementasikan dalam suatu aplikasi yang dibuat dengan bantuan *Integrated Development Environtment (IDE)* Borland Delphi 7. Antar muka hasil implementasi dapat dilihat pada Gambar 3 dan Gambar 4.



Gambar 3. Tampilan Aplikasi untuk proses *Embed*



Gambar 4. Tampilan Aplikasi untuk proses *Extract*

Selanjutnya dilakukan pengujian terhadap aplikasi yang telah dikembangkan. Pengujian yang dilakukan diantaranya : Pengujian penyisipan pesan, pengujian enkripsi *watermark*, pengujian penambahan baris *xref* dan pengujian penyisipan jenis berkas *watermark* yang berbeda.

A. Pengujian penyisipan *watermark*

Tahap pertama dilakukan pengujian terhadap fungsi utama dari aplikasi yang telah dirancang, yaitu mampu melakukan proses penyisipan dan ekstraksi *watermark*. Untuk mengetahui hasilnya maka dilakukan penyisipan beberapa berkas *watermark* pada beberapa berkas *PDF* dengan ukuran bekas dan jumlah baris *xref* yang berbeda. Hasilnya dapat dilihat pada Tabel.1 dan Tabel.2.

Tabel.1 Berkas *PDF* yang akan ditanam *watermark*

No	Berkas *PDF*		
	Nama Berkas	Ukuran Berkas	*Xref* (baris)
1.	Journal_Abraham.pdf	246 KB	302
2.	Journal_Alfian.pdf	6.768 KB	1.575
3.	Journal_Brassil.pdf	69 KB	55
4.	Journal_Horta.pdf	272 KB	107
5.	Journal_Jalil.pdf	301 KB	67

Tabel.2 Berkas *watermark* yang ditanamkan

No	Berkas *Watermark*			
	Sebelum Penyisipan		Hasil Ekstraksi	
	Nama Berkas	Ukuran Berkas	Nama Berkas	Ukuran Berkas
1.	Test1.txt	30 byte	Result1.txt	30 byte
2.	Abstract.txt	421 byte	Result2.txt	421 byte
3.	Test2.txt	40 byte	Result3.txt	40 byte
4.	Test3.txt	102 byte	Result4.txt	102 byte
5.	Test4.txt	53 byte	Result5.txt	53 byte

B. Pengujian enkripsi *watermark*

Xref section pada suatu berkas PDF berupa *plain text*, sehingga ketika dibuka dengan bantuan aplikasi *Text editor* dapat dipersepsi. Agar *watermark* yang disisipkan tidak mudah dipersepsi maka dalam penelitian ini sebelum *watermark* disisipakn, dilakukan enkripsi terlebih dahulu menggunakan algoritma RC4. Berikut contoh *watermark* yang telah dienkripsi dan disisipkan dalam *xref*.

```
0000000000 65535 f
0000004319 00000 n
0000040360 j,L³Ã n
0000041076 00000 n
0000009381 00000 n
0000014801 >tö°> n
0000040690 YÕQ•  n
0000039901 00000 n
0000014822 00000 n
0000034335 öóè°> n
0000040825 ÿ8ð   n
0000040063 00000 n
0000034357 =‰•©/ n
0000039396 gRC / n
0000040886 00000 n
0000040928 00000 n
0000040960 00007 n
```

C. Pengujian penambaan baris *xref*

Pengujian ini dilakukan untuk menangani penyisipan *watermark* yang memiliki ukuran (jumlah *byte*) melebihi kapasitas *xref* yang tersedia. *Xref section* mempunyai ukuran yang terbatas sehingga bila data *watermark* yang disisipkan melebihi kapasitas, maka akan dilakukan penambahan baris *(additional xref)* sehingga seluruh data *watermark* dapat disisipkan. Pada percobaan ini dilakukan penyisipan berkas *watermark* **abstract.txt** dengan ukuran **421 *byte*** ke dalam berkas **Journal_maxemchuck.PDF** dengan ukuran *xref* 38 baris. Berkas *watermark* yang akan disisipkan membutuhkan 421/5= 85 baris xref. Kapasitas xref yang tersedia 38-1=37. Jadi akan dilakukan penambahan jumlah baris sebanyak 85-37=48 baris. Sehingga sebelum *watermark* disisipkan jumlah entri xref pada berkas **Journal_maxemchuck.PDF** menjadi 38+48=86 baris, dan seluruh data *watermark* dapat disisipkan.

D. Pengujian penyisipan jenis berkas *watermark* yang berbeda

Berkas *watermark* yang disisipkan tidak hanya berupa *text*, tetapi jenis berkas lainnya seperti *image*. Pada percobaan ini dilakukan penyisipan berkas berupa *image* **logoitb.jpg** dengan ukuran **2.418 KB** ke dalam berkas PDF **Journal_Alfian.PDF** dengan ukuran **6.768 KB.**

V. KESIMPULAN

Pada penelitian ini telah berhasil dikembangkan metode penyisipan *watermark* di bagian *generation number xref*. Penyisipan *watermark* dilakukan dengan cara menimpa setiap *byte generation number* dengan data *watermark*. Data *watermark* pada *generation number* disisipkan secara acak dan sebelum disisipkan dilakukan proses enkripsi dengan algoritma RC4. Jika kapasitas *xref* yang tersedia tidak mampu menampung seluruh data *watermark*, maka dilakukan penambahan baris *xref (additional xref)*, sehingga seluruh data *watermark* tetap dapat disisipkan.

Ukuran berkas *PDF* tidak akan bertambah, selama kapasitas *xref* yang tersedia, mampu manampung seluruh data *watermark*. Ukuran berkas *PDF* akan bertambah jika terjadi proses *additional xref*.

Data *watermark* yang disisipkan di bagian *xref* tidak mudah dideteksi dan dihapus dengan *tools* khusus yang biasa digunakan mengahapus *watermark* pada berkas *PDF*. Tetapi jika menggunakan aplikasi *Text Editor*, data *watermark* dapat dipersepsi sehingga, dapat dilakukan penghapusan secara manual dan akan merusak data *watermark* yang disisipkan tersebut. Penanganan penghapusan data *watermark* secara manual merupakan salah satu hal yang dapat dikembangkan pada penelitian selanjutnya.

Daftar Pustaka

[1] Adlib. "PDF/Archive – Portable Document Format/Archive." 2011.

[2] Al Shaikhli, Imad Fakhri, Akram M Zeki, Rusydi H Makarim, and Al-Sakib Khan Pathan. "Protection of Integrity and Ownership of PDF Documents using Invisible Signature." International Conference on Modelling and Simulation. IEEE Computer Society, 2012. 533.

[3] AlAhmad, Mohammad A, Imad Fakhri Alshaikhli, and Amal E Alduwaikh. "A New Fragile Digital Watermarking Technique for a PDF Digital Holy Quran." International Conference on Advanced Computer Science Applications and Technologies. IEEE, 2013. 250-253.

[4] Alakk, Walid, Hussain Al-Ahmad, and Alavi Kunhu. "A New Watermarking Algorithm for Scanned Grey PDF Files Using DWT and Hash Function." International Symposium on Communication Systems, Networks & Digital Sign (CSNDSP). IEEE, 2014. 690-693.

[5] Khanzode, Prachi, Siddharth Ladhake, and Shreya Tank. "Digital Watermarking for Protection of Intellectual Property." International Journal of Computational Engineering & Management, 2011: 8-12.

[6] Kim, Mi-Young, Osmar R Zaiane, and Randy Goebel. "Natural Language Watermarking Based on Syntactic Displacement and Morphological Division." 2010.

[7] Lee, I Shi, and Wen Hsiang Tsai. "A new approach to covert communication via PDF files." Signal Processing, 2010: 557–565.

[8] Lethbridge, Timothy C, and Robert Laganière. Object-Oriented Software Engineering Practical software development using UML and Java Second edition. McGraw-Hill Education, 2005.

[9] Liu, Hongmei, Lei Li, Jian Li, and Jiwu Huang. "Three Novel Algorithms for Hiding Data in PDF Files Based on Incremental Updates." Digital-Forensic and Watermarking. USA: Springer, 2012. 167-180.

[10] Mali, Makarand L, Nitin N Patil, and J B Patil. "Implementation of Text Watermarking Technique Using Natural Language Watermarks." Communication Systems and Network Technologies. IEEE Computer Society, 2013. 482-486.

[11] Maxemchuk, N F, and Steven Low. "Marking Text Document." Image Processing. IEEE, 1997.

[12] Rautiainen, Sami. "A Look at Portable Document Format Vulnerabilities." (Science Direct) 14 (2009).

[13] Solichin, Achmad. "Digital Watermarking untuk Melindungi Informasi Multimedia." Budi Luhur Information Technology (BIT), 2010: 1-8.

[14] Tayan, Omar, Muhamad K Kabir, and Yaser M Alginahi. "A Hybrid Digital-Signature and Zero-Watermarking Approach for Authentication and Protection of Sensitive Electronic Documents." The Scientific World Journal, 2014: 14 pages.

[15] Topkara, Umut, Mercan Topkara, and Mikhail J Atallah. "The Hiding Virtues of Ambiguity: Quantifiably Resilient Watermarking of Natural Language Text through Synonym Substitutions." Multimedia and Scurity. ACM, 2006.

[16] U, Jaseena K, and Anita John. "An Invisible Zero Watermarking Algorithm using Combined Image and Text for Protecting Text Documents." International Journal on Computer Science and Engineering (IJCSE), 2011: 2265-2272.