

Serangan *Second Preimage* pada Skema Fungsi Hash Modifikasi Lin et al. Berbasis Algoritma *SmallPresent*[8]

Fadhillah Novita Kaloka¹⁾, Nadia Paramita Retno Adiaty²⁾

1) Badan Siber dan Sandi Negara, fadhillah.novita@bssn.go.id

2) Politeknik Siber dan Sandi Negara, nadia@poltekssn.ac.id

Abstrak

Skema modifikasi Lin et al. merupakan skema Message Authentication Code (MAC) berbasis block cipher hasil modifikasi dari skema Lin et al. Perbedaan yang terdapat pada skema modifikasi Lin et al. dengan skema yang sebelumnya adalah adanya penambahan nilai salt, counter serta perubahan input pesan yang semula sepanjang n menjadi $w \geq 2n$. Serangan *second preimage* dapat dilakukan menggunakan beberapa metode, diantaranya metode Jia et al. dan metode Liu et al. Pada penelitian ini dilakukan serangan *second preimage* dengan menggunakan metode Jia et al. dan Liu et al. Pencarian pesan *second preimage* dengan metode Jia et al. dilakukan dengan membangkitkan modifikasi pesan sebanyak $2^{(n+1)/2}$ dan ditemukan pesan *second preimage*. Pada metode Liu et al. tidak didapatkan pesan *second preimage* karena pada serangan *second preimage* metode Liu et al. terdapat beberapa tahapan yaitu pencarian pesan *fixed point*, perluasan pesan serta pencarian pesan *second preimage*. Pada tahap pencarian pesan *fixed point* tidak ditemukan pesan yang memenuhi syarat sehingga tahap berikutnya tidak dapat dilakukan.

Kata kunci: MAC, Serangan *second preimage* Jia et al., Serangan *second preimage* Liu et al., Skema modifikasi Lin et al., *SmallPresent*[8]

Abstract

Lin et al. modification scheme is a Message Authentication Code (MAC) scheme based on a block cipher which is a modification of Lin et al. scheme. The differences found in the modified scheme of Lin et al. with the previous scheme are the addition of salt values, counters and changes to the message input which was originally along n to $w \geq 2n$. Second preimage attacks can be carried out using several methods including the method of Jia et al. and the method of Liu et al. In this research, a second preimage attack was carried out using the method of Jia et al. and Liu et al. The search for second preimage messages by the method of Jia et al. was done by generating modification messages as much as $2^{(n+1)/2}$ and found the second preimage message. In the Liu et al. method, second preimage message was not obtained because in the Liu et al. method of second preimage attack there were several stages, namely searching for fixed point messages, expanding messages and searching for second preimage messages. At the fixed point message search stage, no messages met the requirements so that the next stage could not be done.

Keywords: MAC, Modification scheme of Lin et al., Second preimage attack Jia et al., Second preimage attack Liu et al., *SmallPresent*[8]

1. PENDAHULUAN

Fungsi hash merupakan fungsi yang memetakan input dengan panjang sembarang ke suatu output dengan panjang yang tetap. Dua sifat yang harus dipenuhi oleh fungsi hash, yaitu kompresi dan mudah dihitung [1]. Suatu fungsi hash dikatakan aman jika memenuhi *preimage resistant*, *second preimage resistant*, dan *collision resistant* [2]. Fungsi hash terbagi menjadi dua jenis, yaitu Message Authentication Code (MAC) dan Manipulation Detection Codes (MDC). MAC merupakan fungsi hash dengan kunci, sedangkan MDC merupakan fungsi hash tanpa kunci [1].

MAC adalah fungsi hash satu arah dengan tambahan kunci rahasia [3]. Pembangunan algoritma MAC memiliki tiga cara. Pertama, konstruksi MAC berdasarkan algoritma *block cipher*, seperti One key CBC-MAC (OMAC). Kedua, konstruksi MAC berdasarkan fungsi hash kriptografi (HMAC). Ketiga, konstruksi MAC berdasarkan hash secara umum (*universal hashing*) [3].

Pada tahun 2009, Jia et al. melakukan penelitian serangan *distinguishing* dan serangan *second preimage* pada CBC-like MAC. Pada penelitian tersebut dijelaskan bahwa serangan *second preimage* merupakan salah satu serangan terhadap fungsi hash. Pada serangan ini, diketahui nilai pesan (M) dan nilai MAC (T) yang kemudian dapat dibuat pesan palsu (M') yang tidak sama dengan pesan (M) yang memenuhi $H_k(M') = T$. Suatu fungsi hash dinyatakan *second preimage resistance* jika secara komputasi sulit untuk mendapatkan pesan palsu (M') yang memenuhi nilai $H_k(M') = T$ apabila diketahui pesan (M) dan $H_k(M) = T$ [3].

Pada tahun 2010, Liu et al. melakukan serangan multikolisi dengan metode Joux [4] dan serangan *second preimage* dengan teknik perluasan pesan Kelsey et al. [5] terhadap skema milik Lin et al. [6]. Serangan *second preimage* dengan menggunakan teknik perluasan pesan memiliki dua tahap dalam penerapannya. Tahap pertama pada serangan tersebut adalah mencari *fixed point* dan tahap kedua adalah melakukan perluasan pesan. Pada penelitian Liu et al.

diperoleh hasil bahwa skema Lin *et al.* rawan terhadap serangan kolisi, *preimage* dan *second preimage* [6]. Liu *et al.* melakukan modifikasi terhadap skema Lin *et al.* untuk menangani kerentanan tersebut. Modifikasi yang dilakukan, yaitu menambahkan *counter* dan *salt* serta mengubah input menjadi $w \geq 2n$.

Serangan *second preimage* dengan metode Jia *et al.* [3] dan Liu *et al.* [6] memiliki perbedaan tahapan dalam penerapan masing-masing serangan sehingga memiliki kompleksitas yang berbeda. Serangan *second preimage* dengan menggunakan metode Jia *et al.* dilakukan dengan memodifikasi pesan pada blok pertama dan kedua, kemudian dicari nilai *intermediate* yang sama sehingga dapat ditemukan pesan palsu (M') yang tidak sama dengan pesan (M) dan memenuhi $H_k(M') = T$. Sementara serangan *second preimage* dengan metode Liu *et al.* dilakukan dengan mencari pesan *fixed point*, kemudian dilakukan perluasan pesan sehingga diperoleh pesan palsu (M') yang tidak sama dengan pesan (M) dan memenuhi $H_k(M') = T$.

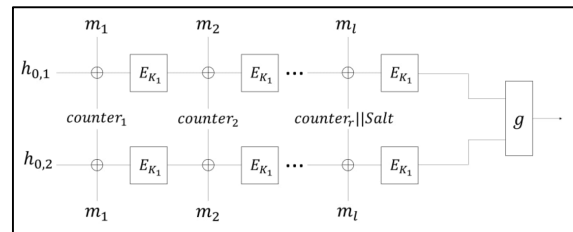
Pada penelitian ini dilakukan serangan *second preimage* pada skema fungsi hash modifikasi Lin *et al.* berbasis algoritma *SmallPresent*[8], dengan menerapkan serangan *second preimage* menggunakan metode Jia *et al.* dan metode Liu *et al.* Penelitian bertujuan untuk mengetahui ketahanan skema fungsi hash modifikasi Lin *et al.* berbasis algoritma *SmallPresent*[8] terhadap serangan *second preimage*.

2. LANDASAN TEORI

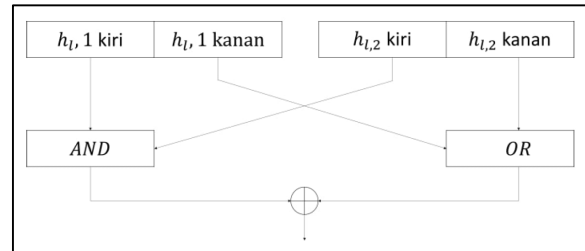
2.1 Skema Modifikasi Lin *et al.*

Pada tahun 2010 Liu *et al.* melakukan serangan terhadap skema Lin *et al.* dengan menerapkan serangan multikolisi yang mencakup serangan kolisi dan serangan *second preimage*, serta melakukan serangan *second preimage* dengan perluasan pesan. Hasil penelitian ini menunjukkan bahwa skema Lin *et al.* rentan terhadap serangan kolisi, *preimage* dan *second preimage* [6]. Liu *et al.* melakukan modifikasi terhadap skema Lin *et al.* dengan menambahkan *counter* dan *salt* serta mengubah panjang bit pada input blok pesan yang semula sepanjang n menjadi $w \geq 2n$. Perubahan panjang input pada skema modifikasi Lin *et al.* bertujuan untuk mencegah ditemukannya kolisi. Konstruksi skema modifikasi Lin *et al.* dapat dilihat pada Gambar 1.

Fungsi g pada skema modifikasi Lin *et al.* memiliki *output* sebesar n dengan panjang input pesan pada tiap bloknya sebesar $w \geq 2n$. Input fungsi g didapatkan dari hasil perhitungan *output* dari cabang B_1 dan B_2 . Nilai $h_{l,1}$ dan $h_{l,2}$ yang didapatkan kemudian dibagi menjadi dua subblok dan dioperasikan dengan fungsi *AND* dan *XOR*. Skema dari fungsi g dapat dilihat pada Gambar 2.



Gambar 1. Skema Modifikasi Lin *et al.*
Sumber: telah diolah dari [6]



Gambar 2. Fungsi g pada Skema Modifikasi Lin *et al.*
Sumber: telah diolah dari [7]

2.2 Algoritma *SmallPresent*[8]

Algoritma *SmallPresent*[m] merupakan modifikasi dari algoritma *Present*[8]. Ukuran blok *SmallPresent*[m] direduksi dari ukuran yang pada awalnya 64 bit menjadi $4m$ bit. *SmallPresent*[m] merupakan algoritma *block cipher* yang menggunakan S-box *Present* sebanyak m dan menggunakan permutasi bit sederhana. Penjadwalan kunci algoritma ini memproduksi $4m$ bit kunci *round* yang dihasilkan dari kunci master sepanjang 80 bit [9].

Algoritma *SmallPresent*[8] dioperasikan dalam tiga tahap yaitu *add round key* dengan melakukan *XOR* antara pesan dengan *round key*, S-box *layer* yaitu substitusi dengan menggunakan S-box *Present*, dan yang terakhir yaitu *permutation layer* untuk melakukan perpindahan posisi. Skema penjadwalan kunci pada algoritma *SmallPresent*[m] hampir sama dengan algoritma *Present*. Perbedaannya terletak pada pengambilan kunci pada setiap *round*. Pengambilan *round key* pada algoritma *SmallPresent*[m] diperoleh dari $4m$ bit paling kanan dari *round key Present*.

2.3 Permasalahan Hari Lahir

Terdapat empat kasus pada permasalahan hari lahir [10]. Permasalahan tersebut berkaitan dengan probabilitas. Probabilitas merupakan nilai yang menyatakan derajat ketidakpastian suatu kejadian pada suatu eksperimen. Asumsi probabilitas ideal suatu kejadian adalah lebih dari sama dengan setengah. Pada penelitian ini, digunakan permasalahan hari lahir keempat yaitu jika diketahui terdapat dua kelas dengan masing-masing kelas beranggotakan sebanyak k siswa, maka tentukan banyaknya jumlah siswa sehingga setidaknya seorang siswa pada kelas pertama memiliki hari lahir yang sama dengan siswa yang berada pada kelas kedua.

Jika diasumsikan terdapat dua himpunan dengan masing-masing himpunan memiliki nilai sebanyak k . Pada permasalahan ini akan ditentukan jumlah minimal k untuk mendapatkan setidaknya ada satu nilai yang sama pada dua himpunan yang berbeda, dengan probabilitas ideal lebih dari sama dengan setengah. Langkah untuk mencari probabilitas pada permasalahan hari lahir keempat adalah sebagai berikut:

- Probabilitas ditemukannya nilai yang berbeda antara sampel pertama pada himpunan pertama dengan himpunan kedua adalah $Q_1 = \left(1 - \frac{1}{N}\right)^k$.
- Probabilitas bahwa semua sampel pada himpunan pertama memiliki nilai yang berbeda dengan himpunan kedua adalah $Q = \left(\left(1 - \frac{1}{N}\right)^k\right)^k$, maka $Q = \left(1 - \frac{1}{N}\right)^{k^2}$.
- Probabilitas ditemukan setidaknya terdapat satu nilai yang sama antara dua himpunan yaitu $P = 1 - Q = 1 - \left(1 - \frac{1}{N}\right)^{k^2}$.

Jumlah minimal k dapat dihitung menggunakan persamaan $k = \sqrt{\ln \frac{1}{1-p} \times N}$. Apabila $P \geq 1/2$, maka jumlah minimal k dapat dihitung menggunakan persamaan $k = 0,83 \times \sqrt{N}$.

2.4 Serangan Second Preimage Metode Jia et al.

Serangan *second preimage* merupakan salah satu serangan terhadap fungsi hash. Pada serangan ini, penyerang memperoleh pesan (M) dan nilai MAC (T) yang memenuhi $H_k(M) = T$. Kemudian, penyerang dapat membuat pesan palsu (M') dengan $M' \neq M$ yang memenuhi $H_k(M') = T$ [3]. Suatu fungsi hash dikatakan *second preimage resistance* jika secara komputasi sulit menemukan pesan palsu (M') yang memenuhi nilai $H_k(M') = T$. Langkah-langkah penerapan serangan *second preimage* dijelaskan sebagai berikut:

- Asumsikan penyerang memiliki nilai hash dari pesan (M), yaitu $H_k(M) = T$ dengan $M = m_1 \parallel m_2 \parallel \dots \parallel m_l$.
- Modifikasi pesan m_1 dan m_2 sebanyak $2^{(n+1)/2}$ untuk membangkitkan himpunan S_1 dan S_2 , dengan $S_1 = \{m_1 \parallel m_2^i \parallel \dots \parallel m_l\}$, dan $S_2 = \{m_1^j \parallel m_2 \parallel \dots \parallel m_l\}$, dengan $m_2^i \neq m_2$ dan $m_1^j \neq m_1$.
- Hitung nilai MAC dari himpunan S_1 dan S_2 , yaitu $T_1^i = H_k(S_1^i)$ dan $T_2^j = H_k(S_2^j)$.
- Pesan palsu dapat ditemukan jika terdapat $T_i = T_j$ sehingga pesan palsu $M' = \{m_1^j \parallel m_2^i \parallel \dots \parallel m_l\}$ memiliki nilai MAC yang sama dengan nilai MAC dari pesan (M).

2.5 Serangan Second Preimage Metode Liu et al.

Teknik perluasan pesan merupakan serangan yang diusulkan oleh Kelsey dan Schneier [5].

Serangan ini memanfaatkan penambahan blok pesan untuk mendapatkan nilai hash yang sama. Teknik ini lebih efisien digunakan jika sudah diperoleh pesan *fixed point*.

Pada tahun 2010, Liu et al. melakukan serangan terhadap skema Lin et al. menggunakan serangan *second preimage* dengan perluasan pesan. Terdapat dua tahap dalam melakukan serangan *second preimage* dengan metode Liu et al., yaitu teknik pencarian pesan *fixed point* dan teknik perluasan pesan secara umum. Teknik pencarian pesan *fixed point* dilakukan pada fungsi kompresi f dengan mencari pasangan IV dan pesan yang memenuhi $H_k = f(h, M)$. Penggunaan *fixed point* dapat memperluas pesan ke sembarang panjang blok tanpa mengubah nilai $H_k(M)$ [6]. Proses ini memiliki tujuan untuk memperbanyak kemungkinan pesan yang menghasilkan nilai *intermediate* yang sama [6]. Langkah-langkah penerapan serangan *second preimage* dengan metode Liu et al. adalah sebagai berikut [6]:

- Mencari pesan *fixed point*
 - Temukan $2^{n/2}$ pasang $(h_{l,1}, m)$ dengan $h_{l,1} = f(h_{l,1}, m)$ menggunakan algoritma *fixed point*, kemudian simpan nilai yang didapatkan pada *List A* = $(h_{0,1}, m)$.
 - Hitung $h'_{l,1} = f(h_{0,1}, m')$ sebanyak $2^{n/2}$ perhitungan. Simpan hasil perhitungan pada *List B* = $(h_{0,1}, m')$.
 - Temukan nilai yang sama antara *List B* dan *List A* yaitu ketika terdapat nilai yang memenuhi $h_{l,1} = h'_{l,1}$.
 - Simpan pesan $(m' \parallel m)$.
- Proses perluasan pesan
 - Hitung $h = f(h_{0,1}, q)$ sebanyak $r - 1$ perhitungan.
 - Hitung $h'_{l,1} = f(h_{0,1}, m')$ sebanyak $2^{n/2}$ perhitungan. Simpan hasil perhitungan pada *List A* = $(h_{0,1}, m')$.
 - Hitung sebanyak $2^{n/2}$ pasang (h', m') dengan $h' = f(h_{tmp}, m')$. Simpan hasil perhitungan pada *List B* = (h', m') .
 - Temukan nilai yang sama antara *List B* dan *List A* sedemikian hingga diperoleh nilai yang menghasilkan $h_{0,1} = h'_{l,1}$. Dapat diperoleh pesan yang berkolisi yaitu $(m', q \parallel q \parallel \dots \parallel m)$.
- Proses serangan *second preimage*
 - Gunakan data yang diperoleh pada proses pencarian *fixed point* yaitu $(m \parallel m')$. Notasikan hasil yang diperoleh tersebut dengan $M_{test} = (m \parallel m')$.
 - Gunakan data yang diperoleh dari proses perluasan pesan yaitu $(m', q \parallel q \parallel \dots \parallel m)$. Notasikan dengan M^* dan nilai *intermediate* dari pesan tersebut dinotasikan sebagai h^* , dengan $h^* = f(h_{0,1}, M^*)$.
 - Temukan M_{link} dengan $h_{i,1} = f(h^*, M_{link})$.

- 4) Untuk j , $0 \leq j \leq 2^r - 1$.
- (a) *Output second preimage* yang diperoleh adalah $M = M_{test} \parallel M^* \parallel M_{link} \parallel M_{i+1} \parallel M_{i+2} \parallel \dots \parallel M_{2^r+r+1}$.
- (b) $M_{test} = M_{test} \parallel M'$.

Pada langkah terakhir diperoleh 2^n pesan palsu pada cabang pertama. Jika ditemukan paling tidak satu pesan palsu pada cabang kedua, maka ditemukan pesan palsu (M') yang memenuhi $H_k(M) = H_k(M')$.

3. METODE PENELITIAN

Serangan *second preimage* dengan metode Jia et al. pada skema modifikasi Lin et al. menggunakan sebanyak 16 sampel pesan untuk mencari pesan palsu yang ditunjukkan pada Tabel 1.

Tabel 1. Sampel Pesan pada Skema Modifikasi Lin et al.

Sampel	m_1	m_2	T
1	0x44444444	0x88888888	0x4468
2	0xeeeeeeee	0xbbbbbbbb	0xc163
3	0x92929292	0xd2d2d2d2	0x4d85
4	0x10101010	0x65656565	0xce7f
5	0x44444444	0x43756347	0x46e2
6	0xeeeeeeee	0xbaca6495	0xde6e
7	0x92929292	0x76cab473	0x5afd
8	0x10101010	0x6bc902ef	0x1e98
9	0xba6643bc	0x88888888	0x52ca
10	0xffe7349b	0xbbbbbbbb	0x2252
11	0x915ceab4	0xd2d2d2d2	0x5ea2
12	0xe9aba955	0x65656565	0x92cf
13	0x675cba73	0xef459075	0x4204
14	0x3253cabf	0xa90c64ba	0xec87
15	0x24789551	0x4567bcdcd	0x5740
16	0xbe435caf	0xee3435ca	0xaec4

Tahapan serangan *second preimage* dengan metode Jia et al. sebagai berikut:

- Bangkitkan 2^{17} pesan palsu pada blok kedua sehingga $S_1 = \{m_1 \parallel m_2^i \parallel \dots \parallel m_l\}$ dengan $0 < i < 2^{17}$. Kemudian hitung $H_k(S_1^i) = T_1^i$.
- Bangkitkan 2^{17} pesan palsu pada blok pertama sehingga $S_2 = \{m_1^j \parallel m_2 \parallel \dots \parallel m_l\}$ dengan $0 < j < 2^{17}$ kemudian hitung $H_k(S_2^j) = T_2^j$.
- Bandingkan nilai T_1^i dengan T_2^j . Jika diperoleh nilai yang memenuhi $T_1^i = T_2^j$, maka simpan nilai m_1^j dan m_2^i . Konstruksi $M' = \{m_1^j \parallel m_2^i \parallel \dots \parallel m_l\}$ kemudian hitung $H_k(M') = T'$. Pesan palsu diperoleh apabila terdapat nilai $T' = T$ dengan $M' \neq M$.

Serangan *second preimage* dengan metode Liu et al. dilakukan pada fungsi kompresi f . Pada skema fungsi hash modifikasi Lin et al. terdapat dua cabang pada fungsi kompresi yang digunakan, yaitu cabang atas (B_1) dan (B_2). Pesan dibagi ke dalam beberapa blok dengan $h_{0,1}$ merupakan IV pada cabang B_1 dan $h_{0,2}$ pada cabang B_2 . Proses serangan dilakukan pada cabang B_1 terlebih dahulu untuk mencari seluruh kemungkinan pesan palsu yang menghasilkan nilai *intermediate hash* yang sama pada cabang B_1 . Kemudian jika hasil sudah didapatkan maka

dilanjutkan dengan proses pencarian pesan palsu pada cabang B_2 . Pesan *second preimage* ditemukan jika pada cabang B_2 juga diperoleh nilai *intermediate* yang sama dengan input yang berbeda.

Fungsi kompresi yang digunakan adalah algoritma *SmallPresent*[8], yang mengoperasikan input sebanyak 32 bit pada tiap blok. Nilai *counter* berupa iterasi dengan panjang 32 bit dan pada blok pesan terakhir nilai *counter* sepanjang 16 bit digabung dengan nilai *salt* sepanjang 16 bit. Nilai *counter* dan nilai *salt* yang digunakan adalah sebagai berikut:

- $counter_1 = 0x00000001$,
- $counter_2 = 0x0002$,
- $Salt = 0x29d1$.

Kunci master yang digunakan pada algoritma *SmallPresent*[8] adalah K_1 dan K_2 sepanjang 80 bit. Nilai kunci master yang digunakan adalah

- $K_1 = 0x796de379adf824c90178$
- $K_2 = 0x53bc06ad8ebc90a24172$.

Berikut adalah tahapan serangan *second preimage* dengan metode Liu et al.:

- Proses *fixed point*
 - Temukan 2^{16} pasang $(h_{l,1}, m)$ dengan $h_{l,1} = f(h_{l,1}, m)$ menggunakan algoritma *fixed point*. Simpan pasangan yang ditemukan pada *List A* = $(h_{0,1}, M)$.
 - Hitung sebanyak $2^{n/2}$ perhitungan $h'_{l,1} = f(h_{0,1}, m')$. Simpan hasil perhitungan pada *List B* = $(h_{0,1}, m')$.
 - Bandingkan nilai pada *List B* dan *List A* sedemikian hingga $h_{l,1} = f(h_{l,1}, m)$ yang memiliki nilai yang sama dengan $h'_{l,1} = f(h_{0,1}, m')$.
 - Apabila ditemukan nilai $h_{0,1} = h'_{l,1}$ yang sama antara *List A* dan *List B*, maka simpan pesan $(m' \parallel m)$ yang bersesuaian.
- Proses perluasan pesan
 - Hitung sebanyak $r - 1$ perhitungan $h = f(h_{0,1}, q)$.
 - Hitung $h'_{l,1} = f(h_{0,1}, m')$ sebanyak 2^{16} perhitungan. Simpan hasil perhitungan pada *List A* = $(h_{0,1}, m')$.
 - Hitung sebanyak 2^{16} pasang (h', m') dengan $h' = f(h_{tmp}, m')$. Kemudian simpan hasil perhitungan pada *List B* = (h', m') .
 - Temukan nilai yang sama antara *List B* dan *List A* yang memenuhi $h_{0,1} = h'_{l,1}$. Sehingga diperoleh pesan yang berkolisi yaitu $(m', q \parallel q \parallel \dots \parallel m)$.
- Proses pencarian pesan *second preimage*
 - Pencarian pesan palsu dilakukan dengan menggunakan data yang diperoleh pada pencarian pesan *fixed point*. Notasikan $M_{test} = (M' \parallel M)$ dan nilai *intermediate* dari pesan tersebut dinotasikan sebagai h_{test}

- dengan $h_{test} = f(h_{0,1}, M_{test})$.
- 2) Gunakan hasil yang diperoleh dari proses perluasan pesan yang dinotasikan dengan M^* dan nilai *intermediate* dari pesan tersebut dinotasikan sebagai h^* dengan $h^* = f(h_{0,1}, M^*)$.
 - 3) Cari input pesan M_{link} sedemikian hingga $h_{i,1} = f(h^*, M_{link})$.
 - 4) Untuk j , $0 \leq j \leq 2^r - 1$.
 - 5) *Output second preimage* yang diperoleh adalah $M = M_{test} \parallel M^* \parallel M_{link} \parallel M_{i+1} \parallel M_{i+2} \parallel \dots \parallel M_{2^r+r+1}$.
 - 6) $M_{test} = M_{test} \parallel M'$.

Pada langkah terakhir diperoleh pesan palsu pada cabang pertama. Jika paling tidak terdapat satu pesan palsu pada cabang kedua, maka ditemukan pesan palsu (M') yang memenuhi $H_k(M) = H_k(M')$.

4. HASIL DAN PEMBAHASAN

4.1 Hasil Serangan *Second Preimage* Metode Jia *et al.*

Serangan ini dilakukan dengan pembangkitan pesan secara acak. Berikut merupakan tahapan yang dilakukan dan hasil yang didapatkan untuk mencari pesan palsu.

- a. Pembangkitan input pesan m'_2 sebanyak $2^{(n+1)/2}$. Pada tahap ini dilakukan pembangkitan pesan palsu pada blok kedua sebanyak 2^{17} pesan atau lebih tepatnya 131.072 pesan palsu. Hitung nilai *hash* yang diperoleh, kemudian simpan sebagai T_1 .
- b. Pembangkitan input pesan m'_1 sebanyak $2^{(n+1)/2}$. Pada tahap ini dilakukan pembangkitan pesan palsu pada blok pertama sebanyak 2^{17} pesan atau lebih tepatnya 131.072 pesan palsu. Hitung nilai *hash* yang diperoleh.
- c. Membandingkan nilai *intermediate* T_1 dan T_2 . Pada tahap ini ditunjukkan apakah terdapat nilai *intermediate* T_1 dan T_2 dengan nilai yang sama. Jika diperoleh nilai *intermediate* yang sama, maka langkah yang selanjutnya dilakukan yaitu dengan melakukan konstruksi pesan baru $M' = M[1]^j \parallel M[2]^i \parallel \dots \parallel M[l]$. Apabila tidak didapatkan nilai yang memenuhi, maka langkah berhenti dan tidak diperoleh pesan yang bersifat *second preimage*. Setelah diperoleh pesan M' kemudian hitung nilai *hash* dari pesan M' . Jika diperoleh nilai *hash* dari pesan M' yang sama dengan nilai T , maka pesan palsu ditemukan.

Secara ringkas, untuk mencari *second preimage* dengan metode Jia *et al.* diperlukan empat langkah sebagaimana yang ditunjukkan pada Algoritma 1. Algoritma 1 menghasilkan *output* yaitu pesan $M' = (m'_1 \parallel m'_2)$ yang memiliki nilai $T' = T$. Pesan palsu didapatkan ketika ditemukan pesan yang memiliki nilai MAC yang sama pada pesan asli. Kolisi tersebut

menyebabkan nilai MAC pesan palsu sama dengan nilai MAC pesan asli. Pesan palsu tersebut yang dinamakan dengan *second preimage*. Pesan palsu yang ditemukan pada suatu pesan asli memungkinkan berjumlah lebih dari satu buah. Hasil *second preimage* yang diperoleh ditunjukkan pada Tabel 2.

Algoritma 1. Serangan *Second Preimage* metode Jia *et al.*

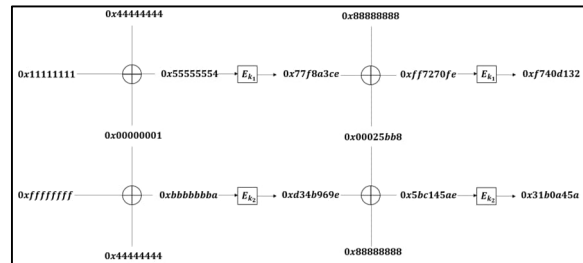
INPUT : nilai MAC $T = H_K(M)$ dengan $M = (m_1 \parallel m_2)$

OUTPUT : $M' = (m'_1 \parallel m'_2)$ dengan $H_K(M') = T$

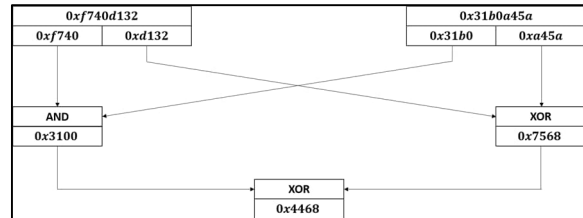
Untuk $i = 0$ sampai dengan $2^{n+1/2} - 1$ lakukan

1. Modifikasi pesan m_2 kemudian hitung nilai MAC $H_K(m_1 \parallel m'_2) = T_1$.
2. Modifikasi pesan m_1 kemudian hitung nilai MAC $H_K(m'_1 \parallel m_2) = T_2$.
3. Bandingkan nilai T_1 dan T_2 .
4. Jika ditemukan nilai yang sama, maka konstruksi pesan $M' = (m'_1 \parallel m'_2)$ dan hitung nilai MAC $H_K(M') = T'$.
5. Bandingkan nilai T' dan T .
6. Jika ditemukan nilai yang sama, maka didapatkan pesan palsu $M' = (m'_1 \parallel m'_2)$.

Tahapan pencarian pesan palsu yang dilakukan dapat diinterpretasikan seperti yang ditunjukkan pada Gambar 3 sampai dengan Gambar 10. Contoh perhitungan pada fungsi f dengan menggunakan pesan asli ditunjukkan pada Gambar 3, serta contoh perhitungan pada fungsi g dengan menggunakan *output* fungsi f dengan input pesan asli ditunjukkan pada Gambar 4.



Gambar 3. Perhitungan pada fungsi f dengan pesan asli



Gambar 4. Perhitungan pada fungsi g dengan pesan asli

Contoh perhitungan pada fungsi f dengan modifikasi pesan pada blok kedua ditunjukkan pada Gambar 5, serta contoh perhitungan pada fungsi g dengan menggunakan *output* fungsi f dengan input modifikasi pesan pada blok kedua ditunjukkan pada Gambar 6.

Contoh perhitungan pada fungsi f dengan modifikasi pesan pada blok pertama ditunjukkan pada

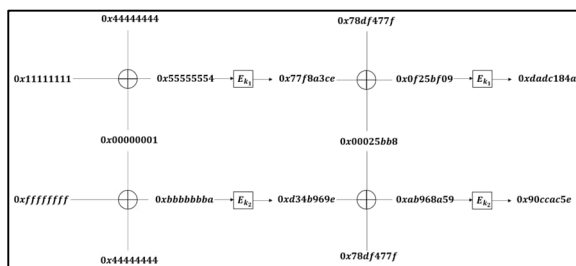
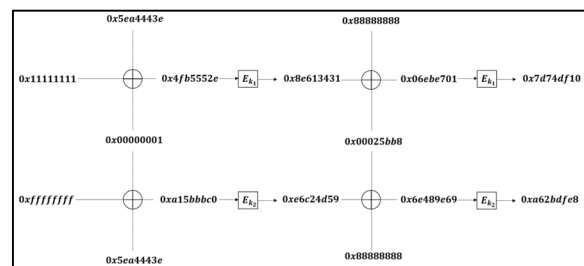
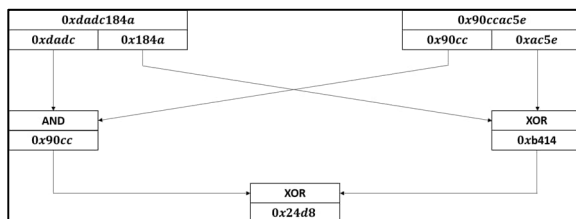
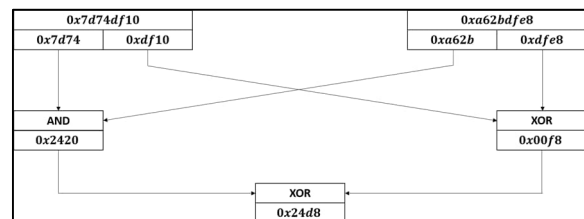
Gambar 7, serta contoh perhitungan pada fungsi g dengan menggunakan *output* fungsi f dengan input modifikasi pesan pada blok kedua ditunjukkan pada Gambar 8.

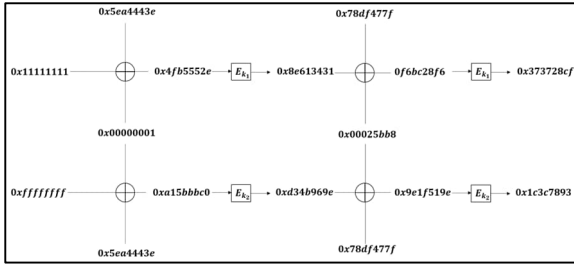
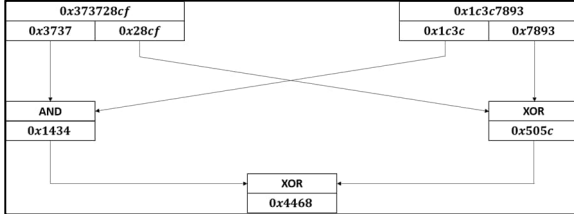
Contoh perhitungan pada fungsi f dengan

menggunakan pesan palsu ditunjukkan pada Gambar 9, serta contoh perhitungan pada fungsi g dengan menggunakan *output* fungsi f dengan input pesan palsu ditunjukkan pada Gambar 10.

Tabel 2. Pasangan m_1' dan m_2' yang menyebabkan nilai $T_1 = T_2$

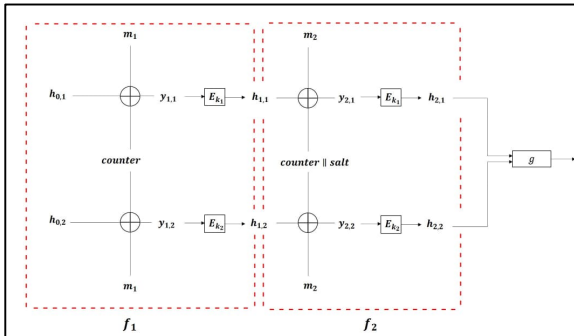
No	m_1	m_1'	m_2	m_2'	$T_1 = T_2$
1	0x44444444	0x5ea4443e 0x0e7e01cf 0x2afb09bc 0x07123297	0x88888888	0x78df477f 0x5dca4368 0x21ed2fc4 0x1bc85da8	0x24d8 0xb055 0x79f6 0x6695
2	0xeeeeeeee	0x1efa228a 0x1bfb58c2 0x494d1382 0x1ac53ab0 0x6737563a 0x278179a3	0xbbbbbbbb	0x3d4a5fb4 0x367a5a53 0x1a1a279f 0x57c43136 0x76fb40ac 0x32247621	0xc95e 0x6ca1 0xde1e 0x56e4 0xaff0 0x47fd
3	0x92929292	0x4b1d71f1 0x75d860c6 0x5bfa009c 0x5c6865bb 0x1c6e3660 0x72676c98	0xd2d2d2d2	0x20a173d4 0x68e4531f 0x3d365262 0x07ad1437 0x46163819 0x13b97221	0xb0bc 0x55d6 0x296a 0xb510 0xb9a0 0xf8fa
4	0x10101010	0x08a84d12 0x668805f1 0x5e1907c3 0x4b667cee	0x65656565	0x07064470 0x2db10630 0x0a692cdb 0x0cc315a1	0x447f 0x90b7 0x60bc 0xea9b
5	0x44444444	0x10ee34a5 0x43587c2b 0x3a051195	0x43756347	0x17723919 0x0d527b49 0xd527b49	0x8fdc 0x2ef4 0x524c
:	:	:	:	:	:
14	0x3253cabf	0x66cd09a2 0x7d3f5a4a 0x1da459c5	0xa90c64ba	0x02225910 0x5cf524fe 0x37d04a87	0x35dd 0x2037 0xbc4f
15	0x47895051	0x72636fde 0x677d4300 0x46567467 0x46567467	0x4567bcdd	0x04e62a1b 0x43a837f0 0x111b1c00 0x192b4569	0x97fe 0x78e9 0x9d1d 0xf3b9
16	0xbe435caf	0x5503753e 0x07f034e3 0x61b5429c 0x07fa6a2e 0x4a5c404f	0xee3435ca	0x23921dfb 0x239b1cc2 0x3a2e6f05 0x30601085 0x6a077256	0x8c15 0xb475 0x082b 0x6f27 0x598f

Gambar 5. Perhitungan pada Fungsi f dengan Modifikasi Blok Pesan KeduaGambar 7. Perhitungan pada Fungsi f dengan Modifikasi Blok Pesan PertamaGambar 6. Perhitungan pada Fungsi g dengan Modifikasi Blok Pesan KeduaGambar 8. Perhitungan pada fungsi g dengan Modifikasi Blok Pesan Pertama

Gambar 9. Perhitungan pada Fungsi f dengan Pesan PalsuGambar 10. Perhitungan pada Fungsi g dengan Pesan Palsu

4.2 Analisis Serangan *Second Preimage Metode Jia et al.*

Skema fungsi hash modifikasi Lin *et al.* menggunakan pesan yang terdiri dari dua blok sehingga $M = m_1 \parallel m_2$. Penerapan serangan *second preimage* dengan metode Jia *et al.* pada skema fungsi hash modifikasi Lin *et al.* didasarkan pada pemanfaatan karakteristik operasi $\oplus$ yaitu ketika $a \oplus b = c \oplus d$ maka $c \oplus b = a \oplus d$ dengan a, b, c , dan d merupakan suatu variabel. Pada skema modifikasi Lin *et al.*, karakteristik tersebut dapat dituliskan sebagai suatu kondisi yaitu ketika $f_1(m_1) \oplus f_2(m'_2) = f(m'_1) \oplus f_2(m_2)$ terpenuhi maka $f_1(m'_1) \oplus f_2(m'_2) = f_1(m_1) \oplus f_2(m_2)$ akan terpenuhi. Fungsi f_1 dan f_2 yang dimaksud ditunjukkan pada Gambar 11.

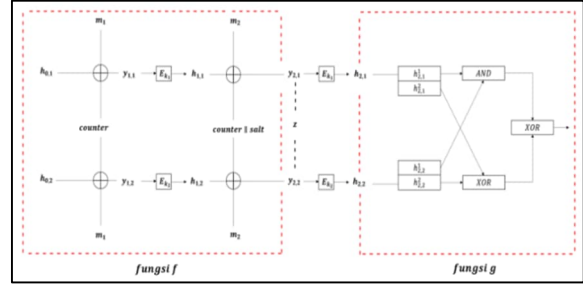
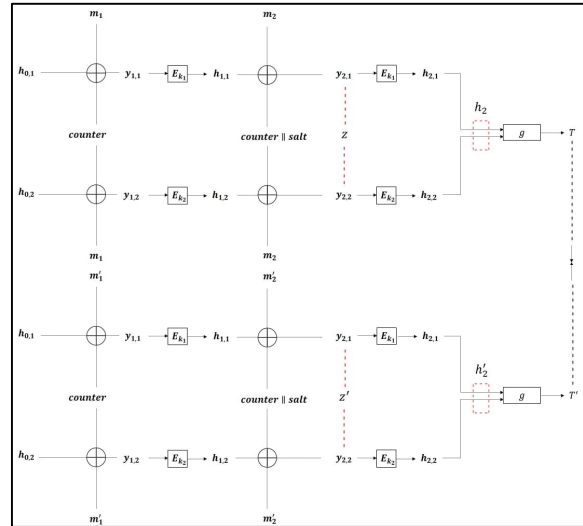
Gambar 11. Fungsi f_1 dan fungsi f_2

Berdasarkan metode Jia *et al.*, pesan palsu berhasil ditemukan apabila langkah-langkah serangan terpenuhi. Langkah-langkah serangan pada skema fungsi hash modifikasi Lin *et al.* dideskripsikan sebagai berikut:

Misalkan serangan *second preimage* dengan metode Jia *et al.* dilakukan terhadap skema fungsi hash modifikasi Lin *et al.* yang terdiri dari dua buah blok input pesan. Diketahui sebarang pesan $M = m_1 \parallel m_2$ dan $f(m_1, m_2) = f(m_1) \oplus f(m_2)$ dengan f merupakan fungsi permutasi sebagaimana

ditunjukkan pada Gambar 12. Jika $f(m_1, m'_2) = f(m'_1, m_2)$ terpenuhi maka $f(m'_1, m'_2) = f(m_1, m_2)$ akan terpenuhi dan jika $f(m'_1, m'_2) = f(m_1, m_2)$ maka $H(m_1 \parallel m_2) = H(m'_1 \parallel m'_2)$.

Pada langkah-langkah serangan, jika $f(m_1, m'_2) = f(m'_1, m_2)$, maka seharusnya terdapat kolisi pada $z = z'$ yang kemudian dapat menghasilkan $H(m_1 \parallel m_2) = H(m'_1 \parallel m'_2)$ seperti ditunjukkan pada Gambar 13.

Gambar 12. Fungsi f dan Fungsi g Gambar 13. Skenario Serangan *Second Preimage Metode Jia et al.*

Pada skema fungsi hash modifikasi Lin *et al.* kondisi sebagaimana pada langkah-langkah yang telah dideskripsikan tidak dapat terpenuhi untuk semua pesan palsu yang ditemukan. Proses pencarian pesan palsu pada skema fungsi hash modifikasi Lin *et al.* secara rinci dapat dilihat pada Gambar 3 sampai dengan Gambar 10.

Pesan hasil modifikasi pada Gambar 5 dan Gambar 7 menunjukkan bahwa tidak terjadi kolisi $z_i = z_j$ pada pesan $M_i = m_1 \parallel m_2^i$ dengan $M_j = m_2^j \parallel m_2$, namun pesan M_i dan M_j dapat direkonstruksi menjadi pesan $M' = m'_1 \parallel m'_2$ yang menghasilkan nilai hash yang sama dengan pesan $M = m_1 \parallel m_2$. Kondisi tersebut kontradiksi dengan strategi serangan *second preimage* dengan metode Jia *et al.* sebagaimana yang dijabarkan pada Gambar 5 dan Gambar 7, menunjukkan bahwa dengan nilai input yang berbeda pada fungsi g antara perhitungan nilai hash M_i dan M_j dapat menghasilkan output T_1 dan T_2 yang sama. Hal ini menunjukkan bahwa strategi

serangan *second preimage* dengan metode Jia *et al.* tidak dapat diterapkan pada skema fungsi hash modifikasi Lin *et al.*

Second preimage pada skema fungsi hash modifikasi Lin *et al.* dapat ditemukan dikarenakan adanya karakteristik operasi $\oplus$ pada fungsi g yang dapat dimanfaatkan untuk mencari kolisi. Berdasarkan Gambar 12, karakteristik operasi $\oplus$ yang digunakan pada fungsi g untuk melakukan penerapan serangan *second preimage* metode Jia *et al.* adalah dengan mencari input $h'_{2,1}$ dan $h'_{2,2}$ yang menghasilkan output fungsi g yang sama dengan input $h_{2,1}$ dan $h_{2,2}$.

Misalnya terdapat output y yang merupakan hasil operasi $\oplus$ dari dua input berbeda, yaitu: x dan x' , dengan $x \neq x'$, dan ukuran n -bit, maka akan terdapat $2n$ kemungkinan pasangan input x dan x' , dengan $x \neq x'$ yang memiliki nilai y yang sama. Misalnya pada sebuah operasi $\oplus$ diketahui nilai output dengan ukuran 4-bit, $y = 0001$, maka kemungkinan nilai input x dan x' yang digunakan ditunjukkan pada Tabel 3.

Tabel 3. Kemungkinan input operasi XOR dengan output 001

No	x	x^*	$y = x \oplus x^*$
1	000	001	001
2	001	000	001
3	010	011	001
4	011	010	001
5	101	100	001
6	100	101	001
7	110	111	001
8	111	110	001

Tabel 3 menunjukkan bahwa nilai output dengan ukuran 4-bit memiliki 24 kemungkinan pasangan input yang menghasilkan nilai $\oplus$ yang sama. Hal tersebut yang menyebabkan terdapat pesan palsu yang memiliki nilai hash yang sama dengan nilai hash pesan asli.

Pencarian pesan *second preimage* pada skema modifikasi Lin *et al.* dilakukan dengan 16 sampel *IV* dan pesan dengan input berpola dan *pseudorandom*. Syarat ditemukannya pesan *second preimage* adalah terdapat pesan M' yang dapat direkonstruksi dengan melakukan modifikasi pada blok pesan pertama dan kedua, serta memenuhi $H_k(M') = H_k(M)$. Proses pencarian pesan *second preimage* dengan metode Jia *et al.* merujuk pada permasalahan hari lahir keempat, yaitu dengan mencari dua nilai yang sama pada dua himpunan yang berbeda.

Berdasarkan hasil yang didapatkan, input *IV* dan pesan, baik berpola maupun *pseudorandom* yang digunakan, tidak mempengaruhi jumlah pesan palsu yang didapatkan. Probabilitas sukses serangan *second preimage* dengan metode Jia *et al.* bergantung pada probabilitas ditemukannya nilai yang sama pada himpunan T_1 dan T_2 .

Himpunan T_1 berisi 2^{17} nilai T dengan modifikasi pada blok pesan kedua. Himpunan T_2 berisi 2^{17} nilai T dengan modifikasi pada blok pesan pertama. Setiap

nilai pada himpunan T_1 dibandingkan dengan nilai pada himpunan T_2 . Apabila ditemukan nilai yang sama antara kedua himpunan tersebut, maka pesan m'_1 dan m'_2 yang bersesuaian digunakan untuk mengganti blok m_1 dan m_2 , kemudian dihitung nilai T' . Apabila terdapat nilai yang sama antara nilai T dan T' maka m'_1 dan m'_2 yang digunakan disebut pesan palsu dengan nilai MAC yang sama dengan nilai MAC dari pesan asli.

Berdasarkan konsep permasalahan hari lahir keempat, probabilitas ditemukannya pesan palsu yaitu $P = 1/N$. Probabilitas ditemukannya nilai MAC pertama pada himpunan T_1 yang memiliki nilai sama dengan nilai MAC pertama pada himpunan T_2 yaitu $P = \frac{1}{2^{32}}$. Q adalah probabilitas nilai MAC pertama pada himpunan T_1 yang memiliki nilai yang berbeda dengan nilai MAC pertama pada himpunan T_2 yaitu $Q = 1 - P = 1 - \frac{1}{2^{32}}$.

Probabilitas nilai MAC pertama pada himpunan T_1 berbeda dengan seluruh nilai MAC pada himpunan T_2 yaitu

$$Q = \left(1 - \frac{1}{2^{32}}\right)^{17} \quad (1)$$

Persamaan (1) dioperasikan dengan pendekatan Taylor ($1 - x = e^{-x}$) menjadi $Q = \left(e^{-\frac{1}{2^{32}}}\right)^{17}$.

Nilai Q juga berlaku untuk nilai MAC kedua, ketiga hingga nilai MAC ke- 2^{16} . Probabilitas seluruh nilai pada himpunan T_1 memiliki nilai yang berbeda dengan nilai pada himpunan T_2 yaitu $Q = 0,018$. Probabilitas ditemukannya paling tidak terdapat satu nilai pada himpunan T_1 yang memiliki nilai yang sama dengan nilai pada himpunan T_2 yaitu $P = 1 - Q = 0,982$.

Berdasarkan probabilitas yang didapatkan, pada percobaan dengan menggunakan 16 sampel, maka setidaknya terdapat 16 pesan palsu yang ditemukan. Berdasarkan hasil yang diperoleh pada Tabel 2, pada masing-masing sampel terdapat pesan palsu yang ditemukan. Hasil tersebut menunjukkan bahwa pada 16 sampel pesan dengan modifikasi pesan sebanyak 2^{17} pada m'_1 dan m'_2 dengan peluang didapatkannya pesan yang sama antara T_1 dan T_2 yaitu sebesar 0,982, seharusnya paling tidak terdapat $15,712 \approx 16$ pesan palsu yang didapatkan.

Serangan *second preimage* dengan menggunakan metode Jia *et al.* memiliki kompleksitas waktu sebesar ($O^{n+3/2}$). Kompleksitas tersebut didapatkan berdasarkan input serta jumlah proses yang dilakukan dalam melakukan serangan. Pada langkah modifikasi pesan m_2 dibutuhkan komputasi sebesar $2^{n+1/2}$. Secara garis besar, serangan *second preimage* metode Jia *et al.* memiliki empat proses untuk menemukan pesan palsu. Kompleksitas waktu yang diperoleh dinotasikan secara formal dengan notasi *Big-O* dengan cara menghilangkan faktor koefisien pada persamaan $T(N)$. Oleh karena $T(N) = 2^{n+3/2} + 2$,

maka kompleksitas waktu serangan *second preimage* yang dilakukan adalah $O(2^{n+3/2})$. Berdasarkan Algoritma 1, serangan *second preimage* dengan metode Jia *et al.* pada penelitian ini dilakukan dengan kompleksitas sebesar $2 \times 2^{17} + 2 = 2^{18}$.

4.3 Hasil Serangan Second Preimage Metode Liu *et al.*

Tahapan pertama pada serangan *second preimage* metode Liu *et al.* yaitu proses pencarian *fixed point* yang dilakukan terlebih dahulu pada cabang pertama. Berikut tahapan yang dilakukan:

- a. Melakukan pencarian pasangan $(h_{0,1}, m)$ sebanyak 2^{16} .

Pada tahap ini dilakukan pencarian pasangan $h_{i,1}$ dan pesan m yang memenuhi $h_{i,1} = f(h_{i,1}, m)$. Pencarian pasangan $(h_{i,1}, m)$ dilakukan dengan 63 percobaan dengan rincian jumlah $h_{i,1}$, m dan $h_{i,1} = f(h_{i,1}, m)$ yang dapat disimpan pada *ListA* dijelaskan pada Tabel 4. Pencarian dilakukan dengan membangkitkan himpunan *IV* dan m , kemudian masing-masing $h_{0,1}$ dan dipasangkan pada semua pesan m , sehingga didapatkan pasangan $(h_{0,1}, m)$, kemudian hitung $f(h_{0,1}, m)$. Apabila ditemukan pasangan yang memenuhi $h_{0,1} = f(h_{0,1}, m)$, maka $(h_{0,1}, m)$ disimpan kedalam *ListA*. Terdapat 133 pasang *IV* dan pesan yang disimpan pada *ListA*, nilai tersebut *ListA* ditunjukkan pada Tabel 5.

Tabel 4. Jumlah $h_{i,1}$, m dan $h_{i,1} = f(h_{i,1}, m)$ yang digunakan

Percobaan ke-	$h_{i,1}$	m	Total pasangan $(h_{i,1}, m)$	$h_{i,1} = f(h_{i,1}, m)$ yang disimpan
1 – 13	2^{16}	2^{16}	2^{32}	11
14 – 25	2^{17}	2^{16}	2^{33}	16
26 – 36	2^{18}	2^{16}	2^{34}	20
37 – 50	2^{14}	2^{20}	2^{34}	34
51 – 63	2^{16}	2^{20}	2^{36}	52

Tabel 5. Initial value dan pesan yang disimpan pada *ListA*

No	$h_{i,1}$	$h_{0,1}$	m
1	0x699f4009	0x699f4009	0x33c9119b
2	0x262d7b25	0x262d7b25	0x3b211bd1
3	0x4eae740a	0x4eae740a	0x349458ac
4	0x267b45ee	0x267b45ee	0x26884699
5	0x06de551b	0x06de551b	0x091877d7
6	0x7a5a11ee	0x7a5a11ee	0x3b665b8a
7	0x6be9578d	0x6be9578d	0x7acf7279
⋮	⋮	⋮	⋮
131	0x5aaf1a57	0x5aaf1a57	0x67b55d60
132	0x6cf4423d	0x6cf4423d	0x68ae0cf7
133	0x1052500b	0x1052500b	0x0a161115

- b. Menghitung $h'_{i,1} = f(h_{0,1}, m')$ sebanyak 2^{16} perhitungan

Pada tahap ini dilakukan perhitungan $h'_{i,1} = f(h_{0,1}, m')$ dengan melakukan modifikasi sebanyak 2^{16} pada m' dengan nilai $h_{0,1}$ yang tetap. Hasil perhitungan $h'_{i,1} = f(h_{0,1}, m')$ yang didapatkan kemudian disimpan pada *ListB*. Nilai yang disimpan pada *ListB* ditunjukkan pada Tabel 6.

- c. Membandingkan nilai pada *ListA* dan *ListB*
Berdasarkan hasil eksperimen yang telah dilakukan, hanya terdapat 133 pasang $(h_{0,1}, m)$ yang dapat disimpan pada *ListA*, sedangkan seharusnya dibutuhkan sebanyak 2^{16} pasang $(h_{0,1}, m)$ yang harus disimpan pada *ListA*. Dari hasil perbandingan *ListA* dan *ListB* yang didapatkan, tidak ditemukan nilai yang sama, sehingga tidak terdapat pesan palsu yang disimpan sebagai $(M' \parallel M)$.

Tabel 6. Initial value dan pesan yang disimpan pada *ListB*

No	$h_{i,1}$	$h'_{0,1}$	m'
1	0xd293bc66	0x11111111	0x00294823
2	0xbb7eb68d		0x18be6784
3	0xaa511f5e		0x4ae13d6c
4	0x6c434aa3		0x2cd672ae
5	0xc6111630		0x69525f90
6	0xbe877a5e		0x16496df1
7	0xe8a0d27d		0x5af141bb
⋮	⋮	⋮	⋮
65534	0xa9829686	0x11111111	0x3a576d20
65535	0x5dedd52d		0x2a5c3f91
65536	0x638b3382		0x702648fa

4.4 Analisis Serangan Second Preimage Metode Liu *et al.*

Pada percobaan yang telah dilakukan hanya didapatkan 133 nilai yang disimpan pada *ListA*, sedangkan menurut Liu *et al.* untuk mendapatkan setidaknya satu nilai yang sama antara *ListA* dan *ListB* maka dibutuhkan sebanyak 2^{16} nilai yang disimpan pada *ListA*. Probabilitas diduplikasinya nilai yang sama pada himpunan *ListA* dan *ListB* berdasarkan percobaan yang telah dilakukan adalah 0,0020978. Misalkan P merupakan probabilitas ditemukannya nilai yang sama antara nilai pertama pada himpunan *ListA* dan nilai pertama pada himpunan *ListB* dan Q adalah probabilitas nilai pertama pada himpunan *ListA* memiliki nilai yang berbeda dengan nilai pertama pada himpunan *ListB*. Proses perhitungan mendapatkan probabilitas P dan Q dijabarkan sebagai berikut $P = \frac{1}{2^{32}}$

Probabilitas nilai pertama pada himpunan *ListA* berbeda dengan seluruh nilai pada himpunan *ListB* yaitu $Q = \left(1 - \frac{1}{2^{32}}\right)^{16}$. Dengan pendekatan Taylor

Series $(1-x = e^{-x})$ menjadi $Q = \left(e^{-\frac{1}{2^{32}}}\right)^{16}$. Nilai Q juga berlaku untuk nilai kedua, ketiga hingga nilai ke-133 pada himpunan *ListA*. Probabilitas seluruh nilai pada himpunan T_1 memiliki nilai yang berbeda dengan nilai pada himpunan T_2 yaitu $Q = 0,9979022$. Probabilitas ditemukannya paling tidak terdapat satu nilai pada himpunan *ListA* yang memiliki nilai yang sama dengan nilai pada himpunan *ListB* yaitu $P = 1 - Q = 0,0020978$.

Misalkan k merupakan jumlah minimal elemen pada himpunan *ListA* sehingga minimal terdapat dua elemen yang sama antara elemen pada himpunan *ListA* dan *ListB* dengan probabilitas $P \geq 1/2$.

Misalkan N merupakan banyaknya kemungkinan nilai *output* dari fungsi kompresi, maka k yang dibutuhkan adalah

$$\begin{aligned}
 P = 1 - Q &\geq 1/2 \\
 1 - \left(\left(1 - \frac{1}{2^{32}} \right)^{2^{16}} \right)^k &\geq 1/2 \\
 1 - e^{-\frac{2^{16}k}{2^{32}}} &\geq 1/2 \\
 -e^{-\frac{k}{2^{16}}} &\geq -\frac{1}{2} \\
 e^{-\frac{k}{2^{16}}} &\leq \frac{1}{2} \\
 e^{\frac{k}{2^{16}}} &\geq 2 \\
 \frac{k}{2^{16}} &\geq \ln 2 \\
 k &\geq \ln 2 \times 2^{16} \\
 k &\geq 45427.
 \end{aligned}$$

Jumlah minimal elemen pada *ListA*, untuk mendapatkan setidaknya satu nilai yang sama antara *ListA* dengan *ListB* dengan probabilitas $P \geq 1/2$ adalah 45427. Pada percobaan yang dilakukan tidak didapatkan nilai yang sama antara *ListA* dengan *ListB* dikarenakan hanya terdapat 133 nilai yang disimpan pada *ListA*, sedangkan untuk mendapatkan nilai yang sama antara *ListA* dan *ListB* dengan probabilitas $P \geq 1/2$, dibutuhkan setidaknya 45427 nilai yang disimpan pada *ListA*.

5. KESIMPULAN

Hasil penelitian ini memiliki kesimpulan bahwa Skema modifikasi Lin *et al.* berbasis *SmallPresent*[8] rentan terhadap serangan *second preimage* menggunakan metode Jia *et al.* karena pada setiap sampel pesan yang digunakan, didapatkan pesan palsu dengan kompleksitas waktu 2^{48} . Strategi serangan *second preimage* dengan metode Jia *et al.* tidak dapat diterapkan pada skema Lin *et al.* Penyebab ditemukannya *second preimage* adalah penggunaan operasi XOR pada fungsi g .

Selain itu, serangan *second preimage* dengan metode Liu *et al.* juga tidak dapat dilakukan pada skema modifikasi Lin *et al.*, dikarenakan pada tahap pencarian pesan *fixed point* tidak didapatkan hasil yang memenuhi syarat. Nilai yang disimpan pada *ListA* hanya terdapat 133, sedangkan seharusnya untuk mendapatkan paling tidak satu nilai yang sama antara *ListA* dengan *ListB* dengan probabilitas $P \geq 1/2$ dibutuhkan sebanyak 45427 nilai yang seharusnya disimpan pada *ListA*.

Beberapa hal berikut dapat menjadi penelitian selanjutnya yaitu melakukan serangan *second preimage* dengan menggunakan metode *chosen plaintext attack* ataupun *chosen message attack* pada skema modifikasi Lin *et al.* dan melakukan serangan *second preimage* dengan menggunakan metode Jia *et al.* pada skema fungsi hash lain.

REFERENSI

- [1] A. J. Menezes, P. C. Van Oorschot, and S. A. Vanstone, *Handbook of applied cryptography*. in CRC Press series on discrete mathematics and its applications. Boca Raton: CRC Press, 1997.
- [2] D. R. Stinson, *Cryptography: Theory and Practice, Third Edition.*, 3rd ed. in Discrete Mathematics and Its Applications. Hoboken: CRC Press, 2005.
- [3] K. Jia, X. Wang, Z. Yuan, and G. Xu, "Distinguishing and Second-Preimage Attacks on CBC-Like MACs," in *Cryptology and Network Security*, J. A. Garay, A. Miyaji, and A. Otsuka, Eds., Berlin, Heidelberg: Springer, 2009, pp. 349–361. doi: 10.1007/978-3-642-10433-6_23.
- [4] A. Joux, "Multicollisions in Iterated Hash Functions. Application to Cascaded Constructions," in *Advances in Cryptology – CRYPTO 2004*, vol. 3152, M. Franklin, Ed., Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 306–316. doi: 10.1007/978-3-540-28628-8_19..
- [5] J. Kelsey and B. Schneier, "Second Preimages on n -Bit Hash Functions for Much Less than $2n$ Work," in *Advances in Cryptology – EUROCRYPT 2005*, vol. 3494, R. Cramer, Ed., Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 474–490. doi: 10.1007/11426639_28.
- [6] B. Liu, Z. Gong, X. Chen, W. Qiu, and D. Zheng, "Cryptanalysis of Lin *et al.*'s Efficient Block-Cipher-Based Hash Function," in *2010 International Conference on Internet Technology and Applications*, Aug. 2010, pp. 1–5. doi: 10.1109/ITAPP.2010.5566639.
- [7] M. N. Salim, "Serangan Collision dan Serangan Fixed Point terhadap Skema Fungsi Hash Modifikasi Lin *et al.* berbasis Algoritma SMALLPRESENT-[8]," Tugas Akhir tidak diterbitkan, Bogor: Sekolah Tinggi Sandi Negara, 2020.
- [8] A. Bogdanov *et al.*, "PRESENT: An Ultra-Lightweight Block Cipher," in *Cryptographic Hardware and Embedded Systems - CHES 2007*, P. Paillier and I. Verbauwhede, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 450–466. doi: 10.1007/978-3-540-74735-2_31. Available: http://link.springer.com/10.1007/978-3-540-74735-2_31.
- [9] G. Leander, "Small Scale Variants Of The Block Cipher PRESENT," *IACR Cryptology ePrint Archive*, vol. 2010, p. 143, Jan. 2010..
- [10] B. A. Forouzan, *Introduction to cryptography and network security*. in McGraw-Hill Forouzan networking series. Boston: McGraw-Hill Higher Education, 2008.