Perbandingan Keamanan Berbagai Algoritma Hash Untuk Digunakan Pada Blockchain

Cut Meurah Rudi - 13514057

***Abstract*—Blockchain menggunakan algoritma hash untuk merepresentasikan seluruh state dari ketika sistem dimulai hingga state saat ini. Pengguna algoritma hash pada blockchain ini merupakan sesuatu yang unik dan orang-orang mempertanyakan sampai kapan fungsi hash tersebut dapat menjaga integritas dari blockchain tersebut. Makalah ini akan membandingkan keamanan dari penggunaan berbagai fungsi hash untuk digunakan mengikat blok-blok pada blockchain. Beberapa fungsi hash yang akan dibandingkan yaitu MD5, SHA-1, SHA-2 dan SHA-3.**

Makalah ini akan membahas keamanan dari fungsi-fungsi hash tersebut dalam kaa dengan penggunaan fungsi tersebut pada blockchain. Beberapa kriteria yang akan diuji cobe untuk fungsi-fungsi hash tersebut yaitu (1) Mencari potensi collision setelah mencapai panjang rantai tertentu, (2) Percobaan penyerang untuk melakukan tampering pada salah blok tanpa mengubah hash akhir, dan (3) Perubahan waktu yang digunakan untuk menghitung nilai hash jika file yang diberikan lebih besar.

Menggunakan makalah ini diharapkan para pengembang blockchain dapat menentukan fungsi hash yang tepat untuk digunakan ketika akan mengembangkan sebuah sistem blockchain dari awal.

I. Pendahuluan

Menurut (Antonopolus, 2013; Tapscott & Tapscott, 2016, Crosby, Nachiappan, Pattanayak, Verma & Kalyanaraman, 2015; UK Government Office for Science 2015) *Blockchain* adalah sebuah basis data terdistribusi yang menyimpan atau merekam data dalam kumpulan yang disebut blok serta saling terhubung dan terikat satu sama lain. Sekumpulan data yang sudah terkumpul disimpan dalam sebuah blok kemudian dikatikan dengan blok sebelumnya dengan cara memasukkan *kode hash* blok sebelumnya ke dalam blok yang baru terbentuk. Proses *hashing* bertujuan untuk mengamankan data pada blok sehingga jika ada perubahan akan mudah diketahui dari blok teratas. *Blockchain* umumnya diterapkan pada sistem *peer-to-peer* dan akan menyimpan kumpulan bloknya masing-masing namun tetap berkomunikasi dengan *node* lain agar setiap kumpulan blok setiap *node* identik.

Terdapat banyak fungsi hash yang dapat digunakan untuk menghasilkan kode hash yang yang mengikat suatu blok dengan blok setelahnya. Makalah ini akan membandingkan beberapa fungsi hash dari sisi aspek keamanan dalam penggunaan fungsi hash tersebut untuk digunakan pada blockchain. Hasil perbandingan ini diharapkan dapat menjadi acuan untuk memiliki fungsi hash mana yang paling tepat untuk digunakan ketika ingin menyimpan data dalam bentuk blockchain.

Cut Meurah Rudi - 13514057 is with Informatics, Institut Teknologi Bandung, Bandung, Indonesia, e-mail: 13514057@std.stei.itb.ac.id (Corresponding author).

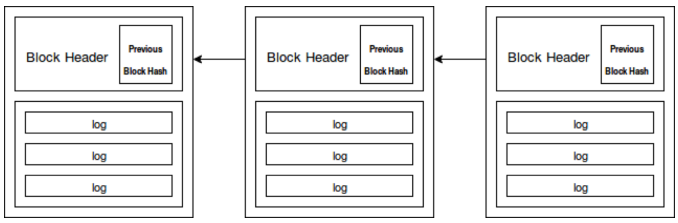

Fig. 1. Susunan Blok pada Blockchain

II. Dasar Teori

A. Blockchain

Pada sebuah sistem *blockchain*, basis data tidak hanya dikelola satu pihak namun dikelola beberapa pihak sehingga basis data bersifat terdistribusi. Untuk memasukkan data baru ke dalam *blockchain* juga tidak dapat dilakukan oleh satu pihak, melainkan harus dilakukan oleh beberapa pihak. Ketika data baru masuk ke dalam *blockchain*, salah satu atau beberapa pengelola *blockchain* akan memvalidasi data tersebut. Apabila data tersebut dinyatakan valid oleh salah satu pengelola, maka data tersebut akan masuk ke dalam daftar tunggu dan disebarkan kepada seluruh pengelola. Pengelola akan melakukan pengecekan terhadap validitas tersebut dan memasukkan ke daftar tunggunya masing-masing jika data yang diterima valid. Setelah daftar tunggu mencapai batas tertentu, pengelola yang terpilih melalui metode konsensus tertentu akan membuat blok baru menggunakan daftar tunggu yang telah diterima. Setelah blok baru ditambahkan pada *blockchain* yang lokal pengelola, blok baru tersebut akan disebarkan ke seluruh jaringan. Pengelola lain akan memvalidasi blok tersebut dan memasukkan ke *blockchain*-nya masing masing apabila blok yang diterima valid.

Misalkan terdapat suatu pihak ingin melakukan transaksi dan ingin menambahkan data transaksi tersebut ke dalam *blockchain*. Maka pihak tersebut akan akan menambahkan data transaksi tersebut ke dalam daftar tunggu miliknya kemudian menyebarkan data transaksi tersebut ke pihak lain. Pihak pengelola *blockchain* lain yang menerima data transaksi akan memasukkan kedalam daftar tunggu miliknya jika data transaksi tersebut valid. Kemudian akan terpilih salah satu pihak melalui prosedur konsensus tertentu untuk membuat block baru dari data transaksi yang telah ada pada daftar tunggu. Blok yang baru terbentuk tersebut akan di beri timestamp dan *hash* blck sebelumnya. Blok yang baru terbentuk tersebut dikirim ke seluruh pihak yang mengelola *node* melalui jaringan. Setiap pihak yang mengelola *node* akan melakukan pengecekan validatas dari blok baru tersebut, jika blok tersebut valid, setiap pihak akan menambah blok tersebut ke *blockchain*

yang dikelolanya. Berikut diagram kerja sederhana *blockchain* tersebut.

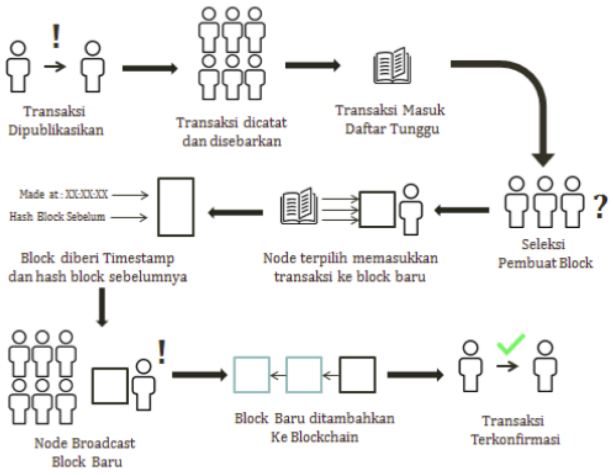

Fig. 2. Diagram Kerja Sederhana Blockchain

Setelah blok baru tersebut bergabung ke dalam *blockchain*, maka data transaksi yang tersimpan akan bersifat semi-permanen. Apabila ada salah satu pihak yang melakukan perubahan pada sebuah blok maka pihak lain akan mengetahuinya melalui kode *hash* blok terakhir yang ditambahkan pada *blockchain*.

B. Konsensus Proof of Work

Konsensus *Proof-of-work* adalah konsensus yang digunakan yang digunakan Bitcoin untuk mengimplementasikan *Timestamp* Server pada basis jaringan *peer-to-peer*. Pada konsensus *proof-of-work* sebuah *node* harus melakukan iterasi hingga mendapatkan kode *hash* yang dimulai dengan beberapa digit nol, tergantung dari kesulitan *proof-of-work* yang ditetapkan. Tingkat kompleksitas dari pekerjaan untuk mendapatkan kode *hash* yang tepat meningkat secara exponensial bergantung pada jumlah *bits* nol yang diinginkan.

Pada jaringan *timestamp* server Bitcoin, *proof-of-work* diimplementasikan dengan meningkatkan sebuah *nonce* pada blok sampai diperoleh nilai kode *hash* yang memiliki kode *bit* nol sesuai jumlah yang diharuskan. Ketika sebuah *node* telah berhasil menjalanakan mekanisme *proof-of-work*, blok tidak dapat diubah kecuali dengan melakukan proof-of-work secara *reversal*. Setelah blok-blok berkaitan menjadi *blockchain*, perubahan blok harus dilakukan dengan melakukan *proof-of-work* secara *reversal* pada blok-blok yang bertautan.

Penggunaan konsensus *proof-of-work* juga menyelesaikan permasalahan pembuatan keputusan dalam sebuah sistem terdistribusi. Jika mayoritas pada sebuah sistem terdistribusi ditentukan dari *one-IP-address-one-vote*, sistem akan lebih mudah dicurangi dengan pihak-pihak yang dapat menghasilkan banyak IP. Menggunakan konsensus *proof-of-work*, mayoritas dipilah berdasarkan *one-CPU-one-vote*. Jika mayoritas CPU dikuasai oleh CPU yang jujur maka rantai blok yang jujur akan berkembang lebih cepat dibandingkan rantai node yang mencoba untuk curang. Untuk melakukan kecurangan pada rantai blok, pihak-pihak curang harus melakukan *proof-of-work reversa*l pada blok-blok yang sudah ditambahkan kemudian membuat rantai blok lain yang harus lebih panjang dari rantai blok yang jujur.

Untuk mengatasi permasalah kecurangan akibat kecepatan *hardwar*e, kesulitan untuk melakukan *proof-of-work* akan bertambah seiring dengan cepatnya pertumbuhan rantai blok.

C. Fungsi Hash

Fungsi hash adalah fungsi yang menerima masukan string yangpanjangnya sembarang dan mengkonversinya menjadi string keluaran yangpanjangnya tetap (fixed). Fungsi hash yang dihasilkan biasanya dituliskan dalamnotasi persamaan sebagai berikut :

$$h = H(M)$$

Pada persamaan di atas, h merupakan nilai hash yang dihasilkan, H adalah fungsi hash itu sendiri, dan M adalah message atau pesan yang akan diubah dandikonversikan menjadi nilai hash (hash value).

Suatu fungsi dikatakan sebuah fungsi hash jika memiliki sifat-sifat sebagai berikut:

1) Fungsi H dapat diterapkan pada blok data berukuran berapa saja.
2) H menghasilkan nilai (h) dengan panjang tetap fixed-length ouput).
3) H(x) mudah dihitung untuk setiap nilai x yang diberikan.
4) Untuk setiap h yang dihasilkan, tidak mungkin dikembalikan nilai x sedemikian hingga $H(x) = h$. Itulah sebabnya fungsi H dikatakan fungsi hash satu-arah (one-way hash function)
5) Untuk setiap x yang diberikan, tidak mungkin mencari y ≠ x sedemikian sehingga $H(y) = H(x)$.
6) Tidak mungkin mencari pasangan x dan y sedemikian sehingga $H(x) = H(y)$.

1) MD-5: Algoritma MD5 adalah fungsi hash yang banyak digunakan menghasilkan nilai hash 128-bit. Meskipun MD5 pada awalnya dirancang untuk digunakan sebagai fungsi hash kriptografi, telah ditemukan menderita kerentanan luas. Ini masih dapat digunakan sebagai checksum untuk memverifikasi integritas data, tetapi hanya terhadap korupsi yang tidak disengaja.

Seperti kebanyakan fungsi hash, MD5 bukanlah enkripsi atau encoding. Ini dapat dipecahkan dengan serangan brute force dan menderita kerentanan yang luas seperti yang dijelaskan di bagian keamanan di bawah ini.

MD5 dirancang oleh Ronald Rivest pada tahun 1991 untuk menggantikan fungsi hash MD4 sebelumnya. Singkatan "MD" adalah singkatan dari "Message Digest."

Keamanan MD5 telah sangat dikompromikan, dengan kelemahannya telah dieksploitasi di lapangan, yang paling terkenal oleh malware Flame pada tahun 2012. CMU Software Engineering Institute menganggap MD5 pada dasarnya "cryptographically broken dan tidak cocok untuk digunakan lebih lanjut".

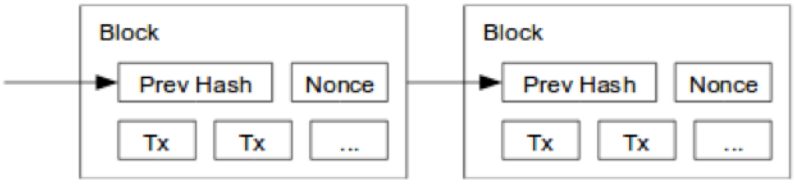

Fig. 3. Setiap blok memiliki nonce yang menentukan kompleksitas proof-of-work

2) SHA-1: SHA-1 (Secure Hash Algorithm 1) adalah fungsi hash kriptografi yang mengambil input dan menghasilkan nilai hash 160-bit (20-byte) yang dikenal sebagai message diges - biasanya diberikan sebagai angka heksadesimal, panjang 40 digit. Ini dirancang oleh National Security Agency (NSA) Amerika Serikat, dan merupakan Standar Pemrosesan Informasi Federal AS.Sejak 2005 SHA-1 belum dianggap aman terhadap lawan yang didanai dengan baik dan sejak 2010 banyak organisasi telah merekomendasikan penggantiannya oleh SHA-2 atau SHA-3. Microsoft, Google, Apple dan Mozilla semuanya telah mengumumkan bahwa browser mereka masing-masing akan berhenti menerima sertifikat SSL SHA-1 pada tahun 2017.Pada 2017 CWI Amsterdam dan Google mengumumkan bahwa mereka telah melakukan serangan tabrakan terhadap SHA-1, menerbitkan dua file PDF berbeda yang menghasilkan hash SHA-1 yang sama.

3) SHA-2: SHA-2 (Secure Hash Algorithm 2) adalah seperangkat fungsi hash kriptografi yang dirancang oleh National Security Agency (NSA) Amerika Serikat. Mereka dibangun menggunakan struktur Merkle-Damgård, dari fungsi kompresi Satu-arah itu sendiri dibangun menggunakan struktur Davies-Meyer dari blok cipher khusus.

Fungsi hash kriptografi adalah operasi matematika yang dijalankan pada data digital; dengan membandingkan computed "hash" (output dari eksekusi algoritma) ke nilai hash yang diketahui dan diharapkan, seseorang dapat menentukan integritas data. Sebagai contoh, menghitung hash file yang diunduh dan membandingkan hasilnya dengan hasil hash yang[1] dipublikasikan sebelumnya dapat menunjukkan apakah unduhan telah dimodifikasi atau dirusak. Salah satu aspek kunci[2] dari fungsi hash kriptografi adalah ketahanan tabrakan mereka:[3] tidak ada yang dapat menemukan dua nilai input yang berbeda yang menghasilkan output hash yang sama. [4]

SHA-2 mencakup perubahan signifikan dari pendahulunya, SHA-1. Keluarga SHA-2 terdiri dari enam fungsi hash dengan mencerna (nilai hash) yaitu 224, 256, 384 atau 512 bit: SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224, SHA - 512/256.

4) SHA-3: SHA-3 (Secure Hash Algorithm 3) adalah anggota terbaru dari standar Secure Hash Algorithm, yang dirilis oleh NIST pada 5 Agustus 2015. Referensi kode sumber implementasi didedikasikan untuk public domain melalui CC0 Waiver. Meskipun bagian dari serangkaian standar yang sama, SHA-3 secara internal cukup berbeda dari struktur mirip MD5 dari SHA-1 dan SHA-2.

SHA-3 adalah bagian dari kriptografi Keccak, yang dirancang oleh Guido Bertoni, Joan Daemen, Michaël Peeters, dan Gilles Van Assche, yang dibangun di atas RadioGatún. Penulis Keccak telah mengusulkan penggunaan tambahan untuk fungsi yang belum distandarisasi oleh NIST, termasuk stream cipher, sistem enkripsi yang diotentikasi, skema hashing "tree" untuk hashing yang lebih cepat pada arsitektur tertentu dan AEAD cipher Keyak dan Ketje.

Keccak didasarkan pada pendekatan baru yang disebut konstruksi spons.Konstruksi spons didasarkan pada fungsi acak yang luas atau permutasi acak, dan memungkinkan memasukkan ("menyerap" dalam terminologi spons) sejumlah data, dan mengeluarkan ("memeras") sejumlah data, sementara bertindak sebagai fungsi pseudorandom berkaitan dengan semua masukan sebelumnya. Ini sangat fleksibel.

NIST saat ini tidak berencana untuk mencabut SHA-2 atau menghapusnya dari Standar Hash Aman yang direvisi. Tujuan SHA-3 adalah dapat secara langsung menggantikan SHA-2 dalam aplikasi saat ini jika diperlukan, dan secara signifikan meningkatkan kekokohan toolkit algoritma hash keseluruhan NIST.

III. Perbandingan

Fungsi hash yang digunakan dalam perbandingan ini yaitu SHA-1, SHA-2, SHA-3 dan MD5. Proses pengambilan nilai hash diperoleh melalui cari berikut ini. Seluruh perbandingan diimplementasikan dalam bahasa java.

```
public static String applySha(String
    input){
  try {
      SHA3.DigestSHA3 digestSHA3 = new
          SHA3.Digest512();
      byte[] hash = digestSHA3.digest(
          input.getBytes());StringBuffer
          hexString = new StringBuffer();
          // This will contain hash as
          hexidecimalfor (int i = 0; i \
          textless\ hash.length; i++) {
          String hex = Integer.toHexString
          (0xff \& hash[i]);if(hex.length
          () == 1) hexString.append('0');
          hexString.append(hex);}return
          hexString.toString();}catch(
          Exception e) {throw new
          RuntimeException(e);}}
```

Code Snippet 1. Fungsi Hash untuk SHA-3

Selain itu blok yang digunakan juga diimplementasikan dalam bahasa java dan diimplementasikan menjadi sebuah kelas. Berikut kelas blok yang digunakan untuk proses percobaan ini.

```
import java.util.Date;public class Block
    {public String hash;public String
    previousHash; private String data;
    private long timeStamp;private int
    nonce;//Block Constructor.  public
    Block(String data,String previousHash
    ) {this.data = data;this.previousHash
    = previousHash;this.timeStamp = new
    Date().getTime();this.hash =
    calculateHash();}//Calculate new hash
    based on blocks contentspublic String
    calculateHash() {String calculatedhash
     = StringUtil.applySha256(
    previousHash +Long.toString(timeStamp)
     +Integer.toString(nonce) + data );
    return calculatedhash;}public void
    mineBlock(int difficulty) {String
    target = StringUtil.getDificultyString
    (difficulty);while(!hash.substring( 0,
     difficulty).equals(target)) {nonce
    ++;hash = calculateHash();}System.out.
    println("Block Mined!!! : " + hash);}}
```

Code Snippet 2. Kelas Blok

A. Waktu Yang Dibutuhkan Untuk Membuat Blok Baru

Pada percobaan ini dicoba waktu yang dibutuhkan untuk membuat blok baru untuk setiap algoritma hash yang digunakan. Percobaan dijalankan pada lingkungan Memory 8 GB 1600 MHz DDR3 dan Processor Intel Core i5 2,6 GHZ dan menggunakan sistem operasi macOS HighSierra.

```
double totalTime = 0;for (int j = 0; j \
    textless\ 10; j++) {long tStart =
    System.currentTimeMillis();addBlock(
    new Block("Hi", "0"));//System.out.
    println("Trying to Mine block 2... ");
    for (int i = 0; i \textless\ 100000; i
    ++) {addBlock(new Block("Hi",
    blockchain.get(blockchain.size() - 1).
    hash));}long tEnd = System.
    currentTimeMillis();long tDelta = tEnd
     - tStart;double elapsedSeconds =
    tDelta / 1000.0;totalTime +=
    elapsedSeconds;System.out.println(
    elapsedSeconds);}System.out.println("
    Average");System.out.println(totalTime
    /10.0);
```

Code Snippet 3. Penghitungan Waktu

Prosedur percobaan yaitu untuk setiap algoritma hash dilakukan percobaan untuk membuat blockchain dengan 100.000 blok kemudian untuk setiap algoritma dilakukan percobaan sebanyak 10 kali sehingga untuk setiap algoritma diperoleh rata-rata waktu yang dibutuhkan untuk membuat 100.000 blok menggunakan setiap algoritma hash.

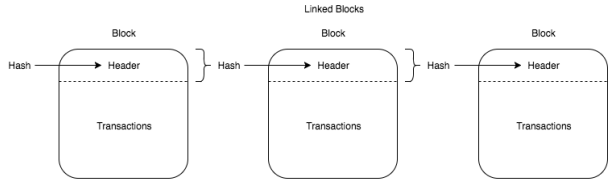

Fig. 4. Susunan Blok Terdiri Dari Header dan Data Transaksi (Sumber : www.pluralsight.com)

Blok yang dibuat dapat dilihat pada Figure 3, setiap blok terdiri dari sebuah Header dan Transactions. Header merupakan nilai hash dari blok sebelumnya sedangkan transaction merupakan sebuah string. Percobaan ini membuat blok sederhana tersebut sepanjang 100.000 blok.

Proses pembuatan blok ini tidak menggunakan sistem peer to peer tetapi hanya mensimulasikan satu buah sistem blockchain yang hanya memiliki satu node.

TABLE I
WAKTU YANG DIBUTUHKAN UNTUK MEMBUAT 100.000 BLOK BARU (DETIK)

No Percobaan	MD5	SHA-1	SHA-2/SHA-256	SHA3-512
1	0.873	0.789	0.866	1.63
2	0.212	0.32	0.521	0.598
3	0.153	0.555	0.638	0.892
4	0.433	0.36	0.303	0.508
5	0.267	0.232	0.409	0.6
6	0.236	0.279	0.753	0.52
7	0.122	0.849	0.376	0.809
8	0.189	0.188	0.291	0.561
9	0.972	0.41	0.706	0.855
10	0.117	0.181	0.268	0.524
Rata-Rata	0.3574	0.4163	0.513	0.7496

B. Percobaan Tampering

Percobaan Tampering dilakukan juga dengan menggunakan rangkaian blok yang dihasilkan pada percobaan sebelumnya. Pada percobaan ini salah satu blok dari 100.000 rangkaian blok yang dihasilkan sebelumnya dicoba untuk diubah data transaksinya tanpa harus mengubah nilai hash blok tersebut. Terdapat dua metode yang digunakan untuk melakukan tampering yaitu menggunakan brute force attack dan dictionary attack.

Anti-tampering merupakan salah satu keunggulan blockchain dimana blockchain diklaim sangat merupakan basis data yang sangat aman karena sangat sulit untuk di-tamper. Tujuan dari percobaan ini untuk menguji klaim tersebut dan menguji keamanan blockchain dari tampering attack.

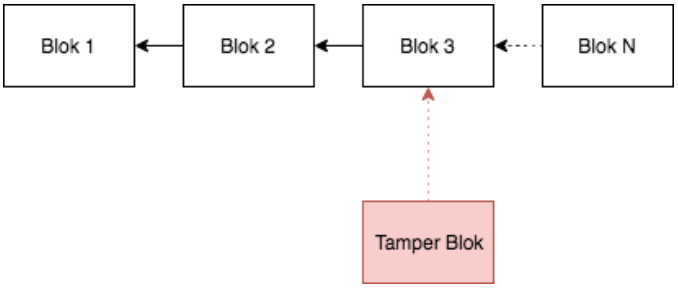

Fig. 5. Skema Percobaan Tampering Attack

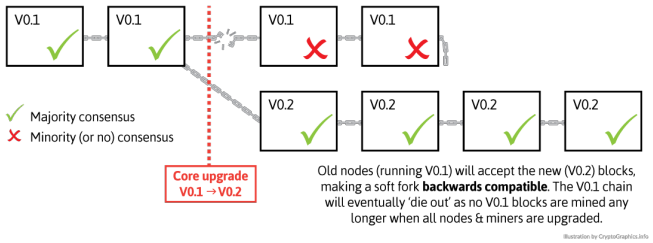

Fig. 6. Skema Percobaan Pembuatan Blockchain Palsu (Sumber : cryptographics.info)

TABLE II
HASIL PERCOBAAN TAMPERING PADA SALAH SATU BLOK

Nama Serangan	MD5	SHA-1	SHA-2/SHA-256	SHA3-512
Brute Force Attack	Tidak Berhasil	Tidak Berhasil	Tidak Berhasil	Tidak Berhasil
Dictionary Attack	Tidak Berhasil	Tidak Berhasil	Tidak Berhasil	Tidak Berhasil

C. *Pencarian Collision*

Percobaan pencarian Collision dilakukan pada satu juta rangkaian blok yang dihasilkan dengan metode pembuatan blok yang dihasilkan pada percobaan pertama yang mencari waktu yang dibutuhkan untuk mencoba blok baru. Collision dikatakan terjadi jika ditemukan dua atau lebih header blok yang memiliki nilai yang sama. Perlu diingat bahwa header blok adalah nilai hash pada blok sebelumnya.

Pada percobaan pencarian collision menggunakan satu juta blok ini tidak ditemukan satu pun collision yang terjadi. Proses pencarian yang lama menyebabkan percobaan sulit jika dilakukan pada blok yang lebih banyak.

TABLE III
PERCOBAAN PENCARIAN COLLISION

MD5	SHA-1	SHA-2/SHA-256	SHA3-512
Collision Tidak Ditemukan	Collision Tidak Ditemukan	Collision Tidak Ditemukan	Collison Tidak Ditemukan

IV. KESIMPULAN DAN SARAN

Berdasarkan percobaan pertama diperoleh bahwa algoritma hash MD5 memiliki waktu yang paling cepat untuk membuat blok baru sedangkan yang kedua tercepat adalah SHA-1, ketiga tercepat adalah SHA-2 dan yang terlama adalah SHA-3. Semakin lama waktu yang dibutuhkan untuk membuat blok maka semakin lama pula waktu yang diperlukan bagi node yang ingin memalsukan blockchain untuk menghasilkan blokchain yang tidak valid.

Pada gambar berikut merupakan skema untuk serangan pada sebuah sistem peer to peer blockchain dimana salah satu node mencoba untuk membuat blockchain palsu. Semakin la

ma waktu yang dibutuhkan untuk membuat blok baru maka semakin kecil pula kemungkinan sebuah node untuk memanipulasi blockchain

Sedangkan pada percobaan kedua proses tampering attack dan dictionary attack gagal dilakukan. Proses ini gagal diperkirakan dikarenakan karena dibutuhkan waktu yang lebih lama dan memori serta kemampuan komputasi yang lebih besar untuk melakukan proses tampering.

Hal yang sama berlaku juga pada pencarian collision pada blok satu juta blok yang sudah dibuat. Hasilnya tidak diperoleh collision sama sekali pada satu juta blok tersebut. Dibutuhkan komputer dengan kemampuan komputasi lebih besar agar dapat melakukan pencarian pada miliaran atau triliunan blok sehingga memperbesar kemungkinan untuk mendapat nilai collision pada sebuah blok header yang berisi nilai hash dari blok sebelumnya.

V. REFERENSI

Antonopoulos, A. M. (2013). *Mastering Bitcoin Unlocking Digital Crypto-currencies.* O Reilly.

Tapscott, A., & Tapscott, D. (2016). *Blockchain Revolution: How the Technology Behind Bitcoin Is Changing Money, Business, and the World.* Portofolio.

Crosby, M., Nachiappan, Pattanayak, P., Verma, S., & Kalyanaraman, V. (2015). *BlockChain Technology Beyond Bitcoin.* Berkeley: Sutardja Center.

Munir, Rinaldi. (2005). Diktat Kuliah Kriptografi. Depatement Bandung : Teknik Informatika, Insitut Teknologi Bandung

Angga, Christian. (2011). Analisis Cara Kerja Beragam Fungsi Hash yang Ada. Bandung: Institut Teknologi Bandung.

Stevens, Marc; Bursztein, Elie; Karpman, Pierre; Albertini, Ange; Markov, Yarik. "The first collision for full SHA-1"

Hernandez, Paul (2015). "NIST Releases SHA-3 Cryptographic Hash Standard"

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 17 Mei 2018

-

Penulis

Cut Meurah Rudi - 13514057 13514057@std.stei.itb.ac.id