Perancangan *Database* IoT Berbasis *Cloud* dengan Restful API

Cloud-Based IoT Database Design with Restful API

Mohammad Kasyful Anwar[1], Tjahjanto[2]
[1,2] Magister Komputer, Universitas Budi Luhur, Jakarta, Indonesia
E-mail: [1]2011600125@student.budiluhur.ac.id, [2]cahyanto@budiluhur.ac.id

Abstrak

Internet of things (IoT) memiliki potensi besar di era industri 4.0, saat mulai muncul beberapa produk elektronik rumah tangga berbasis IoT seperti lampu, saklar, stop-kontak dan lainnya, dengan IoT perangkat rumah tangga tersebut bisa dikendalikan dan dipantau dari jarak jauh sehingga lebih memudahkan penggunanya. Penyimpanan data IoT berbasis *cloud* menimbulkan permasalahan keamanan data dan performa *throughput* pada *server*. Dalam penelitian ini dibahas mengenai rancangan *database* berbasis *cloud* dengan restful API untuk IoT agar data IoT aman dan memiliki *throughput* yang bagus dengan struktur data yang diatur pada *database*.

Kata kunci: *Cloud*, IoT, Restful API, *Database*

Abstract

The Internet of things (IoT) has great potential in the industrial era 4.0, when several IoT-based household electronic products such as lamps, switches, sockets and others began to appear, with IoT these household devices can be controlled and monitored remotely so that more makes it easy for its users. Cloud-based IoT data storage raises issues of data security and throughput performance on servers. This research discusses the design of a cloud-based database with a restful API for IoT so that IoT data is safe and has good throughput with the data structure that is regulated in the database.

Keywords: Cloud, IoT, Restful API, Database

1. PENDAHULUAN

IoT merupakan implementasi internet untuk berbagai benda di dunia nyata seperti perangkat elektronik rumah tangga, peralatan industri, transportasi[1], [2] hingga hewan yang diberikan sensor tertanam dan selalu aktif (untuk keperluan penelitian misalnya). Tujuan IoT adalah agar berbagai benda tersebut bisa terhubung dengan internet[3], [4] sehingga memiliki kemampuan untuk dipantau dan dikendalikan dari jarak jauh, data pada perangkat bisa diakses dan dipelajari dari jarak jauh. Berikut adalah beberapa contoh keuntungan IoT :

- Sektor transportasi, dengan IoT trafik lalu lintas kendaraan di jalan bisa dipantau, jika ada kemacetan bisa langsung diketahui dan dilakukan tindakan lebih lanjut seperti rekayasa lalu lintas. Contoh lain, penggunaan bahan bakar kendaraan bisa dipantau sehingga bisa dianalisa lebih lanjut agar penggunaan bahan bakar bisa lebih hemat.
- Sektor rumah tangga, peralatan rumah tangga seperti lampu, kunci, kompor, pakan hewan[5], *microwave*, kulkas dan lainnya bisa dikendalikan dari jarak jauh bahkan jika pemilik sedang tidak berada di rumah, ini meningkatkan keamanan rumah, selain itu peralatan rumah tangga bisa diketahui data penggunaan listriknya.
- Sektor industri, mesin industri bisa dipantau kinerja dan performa nya, jika terjadi kerusakan bisa terdeteksi dan diberitahukan ke pengguna untuk dilakukan tindakan lebih lanjut seperti perbaikan sebelum kerusakan semakin besar [6].

Perangkat IoT memiliki kapasitas penyimpanan yang relative kecil karena harus menyesuaikan ukuran fisik dari perangkat dan biaya jika menggunakan kapasitas penyimpanan yang besar. Penyimpanan IoT juga tidak memiliki *backup* jika terjadi kerusakan seperti kebakaran atau bencana, jika terjadi seperti itu biasanya perangkat IoT diganti dengan unit baru dan data sebelumnya dibiarkan hilang (karena tidak ada kerugian ekonomis) padahal data tersebut penting untuk dilakukan analisa lebih lanjut untuk pengembangan produk IoT selanjutnya.

Perangkat IoT perlu penyimpanan data berbasis *cloud* agar data bisa tersimpan dengan aman dengan sistem *backup* yang handal dan performa *throughput* yang bagus. Pada penelitian berjudul "*IFCIoT: Integrated Fog Cloud IoT Architectural Paradigm for Future Internet of Things*" integrasi perangkat IoT dengan *cloud* membantu meningkatkan performa dan skabilitas dari perangkat IoT [7]. Pada penelitian berjudul "*A Cloud-IoT Based Sensing Service for Health Monitoring*" perangkat IoT bisa diintegrasikan dengan *cloud* untuk meningkatkan aksesbilitas dan penggunaan data dari sensor perangkat IoT [8].

Pada penelitian berjudul "*IMPLEMENTATION OF FACE SIMILARITY USING TINY FACE DETECTOR*" *Cloud computing* menyediakan komputasi, penyimpanan, layanan dan aplikasi melalui internet [9], dengan *cloud computing* penyimpanan data bisa diakses dengan internet baik itu membaca data (*read*) atau menulis data (*write*) sehingga menjadi solusi penyimpanan IoT karena cukup melalui 1 jalur yang sama yaitu internet. Pada penelitian ini diimplementasikan *relational database* pada *cloud computing* agar data tersimpan secara terstruktur sehingga data bisa dikelola dan dianalisa dengan baik.

Relational database merupakan basis data dimana data disimpan dalam bentuk tabel yang terdiri dari baris dan kolom. *Relational database* memiliki bentuk *schema* (struktur tabel) yang tetap. *Relational database* menggunakan *index* dan *key* sehingga membuat perintah dalam *query* dapat dengan cepat melakukan manipulasi data yang terstruktur seperti penelitian yang telah dilakukan berjudul "Perbandingan Performa *Relational*, *Document-Oriented* dan *Graph Database* Pada Struktur Data *Directed Acyclic Graph*" [10].

Restful API juga diterapkan pada *database* berbasis *cloud* pada penelitian ini agar keamanan komunikasi data tetap terjaga dan menghindari peretasan pada *database* secara langsung. Seluruh perangkat IoT akan berkomunikasi melalui Restful API kemudian data dilanjutkan ke *database* untuk disimpan. Keamanan database merupakan hal utama dalam penelitian ini karena *database* berbasis cloud bisa diakses oleh siapa saja di internet seperti penelitian yang berjudul "Penerapan Keamanan Penggunaan Data pada *Database* Kepegawaian Menggunakan Teknik *Transparent Data Encryption* (Studi Kasus Sekolah Tinggi Teknologi Payakumbuh)" dan penelitian berjudul "Pengamanan Data pada Media Penyimpanan *Cloud* Menggunakan Teknik Enkripsi dan *Secret Sharing*" [11], [12].

REST (*Representional State Transfer*) adalah suatu arsitektur metode komunikasi yang menggunakan protokol HTTP untuk pertukaran data dengan tujuan untuk menjadikan sistem yang memiliki performa yang baik, cepat dan mudah untuk dikembangkan (*scale*) terutama dalam pertukaran dan komunikasi data. Terdapat beberapa metode HTTP yang bisa digunakan pada REST yaitu *GET, POST, PUT* dan *DELETE* [13].

Sistem *web service* yang menerapkan prinsip - prinsip REST disebut RESTful. Cara kerja RESTful yaitu *client* mengirim sebuah permintaan melalui HTTP *Request* dan *server* menanggapi permintaan *client* melalui HTTP *Response*. Terdapat 2 bagian pesan untuk berkomunikasi dengan *server* yaitu pesan *header* dan pesan *body*. HTTP *header* merupakan catatan kecil setiap transaksi pada HTTP sedangkan HTTP *body* yaitu data yang akan dikirimkan [14].

Penelitian mengenai database IoT telah dilakukan sebelumnya dengan judul "Implementasi Teknologi *Internet of Things* Pada Sistem Pemantauan Kebocoran Gas LPG dan Kebakaran Menggunakan *Database* Pada *Google Firebase*" dan "Perbandingan Performa

Database Apache HBase dan *Apache Cassandra* Sebagai Media Penyimpanan Data Sensor *Internet of Things*" [15], [16].

Keterkaitan penelitian ini dengan penelitian-penelitian sebelumnya yaitu pada penelitian sebelumnya, database IoT dirancang menggunakan *Google Firebase* dan pada penelitian lainnya melakukan perbandingan performa *database IoT*. Sedangkan pada penelitian ini database IoT dirancang menggunakan *relational database* dan pengujian keamanan *database* IoT menggunakan uji coba serangan *brute force*.

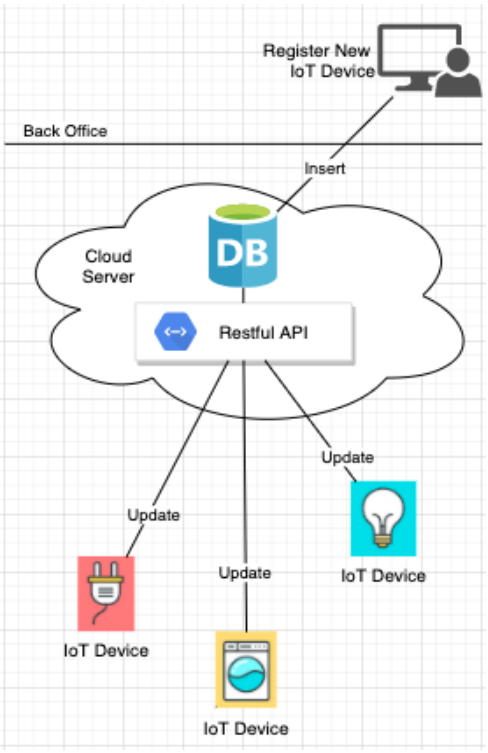
2. METODE PENELITIAN

Existing system

Kondisi saat ini perangkat IoT berjalan masing – masing dengan penyimpanan data sendiri. Data perangkat IoT biasanya dikomunikasikan dengan perangkat pengguna seperti *smartphone* kemudian data tersebut dikirim ke *server cloud*, atau dikirimkan langsung ke *server cloud*. Keamanan data IoT dan *database* penyimpanannya sangat *critical* karena *server cloud* bisa diakses oleh siapa saja di internet [11], [12], struktur data IoT juga harus diatur agar performa *throughput* menjadi maksimal. Dalam penelitian ini dibahas mengenai bagaimana merancang *database* untuk IoT yang aman dengan Restful API sebagai *firewall* atau perantara antara *database* dengan perangkat IoT dan memiliki *throughput* yang bagus dengan struktur data yang diatur pada *database* dan pada saat melakukan komunikasi data dengan Restful API.

Proposed model

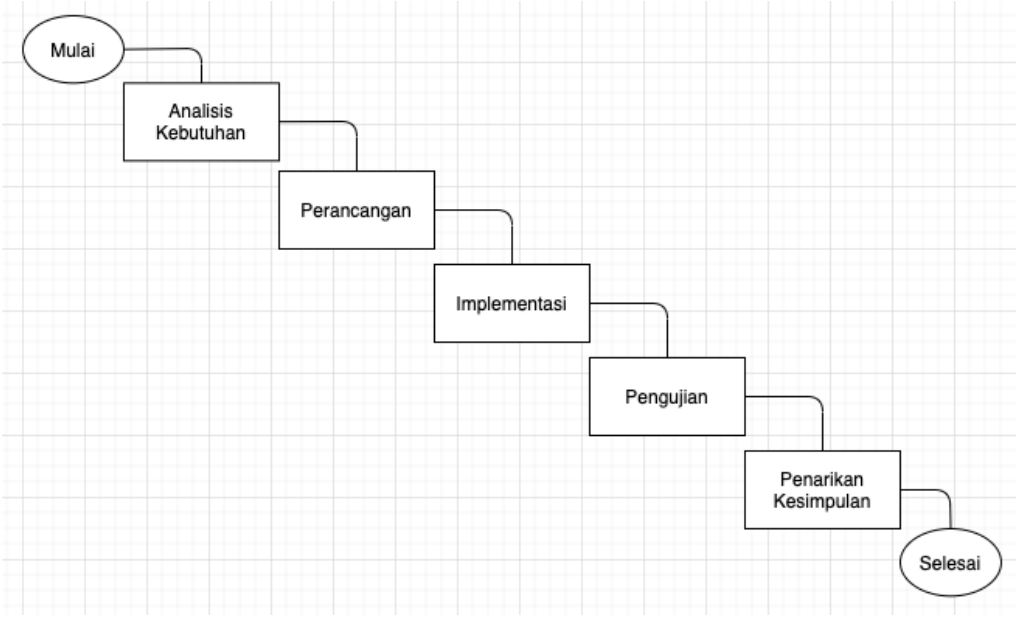
Pada penelitian ini digunakan *relational database* berbasis *cloud* dengan 1 *domain* dan beberapa alamat IP *public* sebagai *backup* apabila terkena serangan *flooding* maka bisa ganti ke IP lainnya serta Restful API sebagai perantara antara *database* dengan lalu lintas internet agar *database* tidak terkena peretasan karena *database* tidak berkomunikasi langsung dengan internet. Restful API juga berfungsi melakukan pengecekan apakah data yang akan dikirim ke *database* adalah data IoT yang *valid* sehingga *database* menjadi lebih aman.



Gambar 1. *Proposed model*

Restful API melakukan pengamanan dari sisi pencegahan komunikasi langsung ke *database* untuk menghindari serangan *brute force* dan lainnya sehingga memastikan *database* menjadi lebih aman. Pengamanan yang dilakukan adalah pencegahan *database* berkemunikasi secara langsung melalui *port* 3306 dengan menutup *port* tersebut dan dilakukan uji coba serangan *brute force* ke *database* untuk melihat hasil dari pengamanan yang dilakukan.

Berikut adalah *flowchart* metodologi penelitian yang digunakan dalam perancangan *database* IoT berbasis *cloud* dengan Restful API :



Gambar 2. Metodologi Penelitian

Analisis Kebutuhan

Menganalisa kebutuhan *database* dan API untuk penyimpanan data yang aman dan tersedia melalui infrastruktur *cloud*.

Perancangan

Perancangan struktur dan desain *database* beserta API untuk melakukan trasmisi atau *update* data beserta rancangan pengamanan nya.

Implementasi

Implementasi *database* dan API yang telah didesain ke *server* dengan melakukan coding SQL untuk *database* dan coding PHP untuk API.

Pengujian

Melakukan pengujian *database* dan API yang sudah diimplementasi pada *server* dengan menggunakan menggunakan metode *Black box Testing* yaitu pengujian yang didasarkan pada detail aplikasi seperti tampilan aplikasi, fungsi-fungsi yang ada pada aplikasi, dan kesesuaian alur fungsi dengan proses yang diinginkan. [14].

Untuk pengujian fungsional dan *fail-over* dilakukan simulasi melalui *client* dengan bantuan aplikasi *Postman*. *Postman* adalah aplikasi yang berfungsi sebagai REST *Client* untuk uji coba REST API. *Postman* digunakan oleh *developer* pembuat API sebagai *tools* untuk menguji API yang telah dibuat. [17]

Untuk pengujian keamanan dilakukan uji coba serangan *brute force* ke *database* IoT dengan menggunakan Hydra di sistem operasi Ubuntu. Hydra adalah salah satu *tools security* yang dipakai untuk melakukan *cracking password* secara *remote*. *Tools* ini dikembangkan oleh THC (*The hacker choice*) di bawah lisensi AGPLv3. [18]

Penarikan Kesimpulan

Penarikan Kesimpulan dilakukan berdasarkan hasil analisa dan pengujian yang telah dilakukan, apakah hasil dari *database* dan API telah memenuhi kebutuhan dan aman digunakan. Kemanan *database* menjadi hal utama dalam penelitian ini.

3. HASIL DAN PEMBAHASAN

Data yang dianalisa dari IoT adalah data lokasi GPS (*Longitude, Latitude*), durasi lama hidup perangkat (dalam menit), dan konsumsi listrik. Data pribadi pengguna seperti nama, alamat email, nomor handphone dan lainnya tidak disimpan di *database* untuk menjaga privasi dari pengguna IoT.

Data lokasi GPS digunakan untuk melihat sebaran perangkat IoT yang ada, ini berguna untuk analisa pemasaran produk IoT tersebut. Data durasi lama hidup perangkat digunakan untuk analisa kualitas dari perangkat IoT apakah setelah dipasarkan dan digunakan oleh *customer* perangkat tersebut memiliki kualitas yang baik atau tidak. Dan data konsumsi listrik digunakan untuk menganalisa seberapa besar energi yang digunakan oleh perangkat IoT tersebut.

Semua data tersebut digunakan untuk analisa dan kemudian dan dilakukan pengembangan lebih lanjut oleh produsen dari produk IoT tersebut seperti membuat produk baru dengan konsumsi energi yang lebih efisien atau produk baru dengan kualitas yang bagus dan sebagainya.

Data tersebut juga bisa diintegrasikan dengan aplikasi *Google Maps* untuk melihat sebaran pengguna produk IoT dan menjadi tolak ukur untuk melihat hasil pemasaran di area tertentu dan untuk melakukan pemasaran di area yang belum menggunakan produk IoT.

Data dikirimkan ke *server cloud* setiap perangkat baru dinyalakan dan cukup 1 kali pengiriman agar lalu lintas data ke *server cloud* tidak terlalu padat dengan *syntax query UPDATE* sehingga hanya *serial number* dari IoT terdaftar saja yang bisa melakukan komunikasi dan menyimpan data ke *database*. Sebelumnya setiap perangkat IoT dimasukkan ke *database* (*INSERT*) setelah selesai proses produksi sehingga produk IoT tersebut terdaftar dan bisa berkomunikasi dengan *database*. Ini sebagai upaya pengecekan dan *valid*asi agar *server cloud* dan *database* tetap tersedia dengan baik dan terhindar dari peretasan. Ini bentuk pengamanan dari sisi *database*.

Spesifikasi yang digunakan untuk *database cloud* :

- *Server cloud* dengan kecepatan internet *Up to* 1 Gbps.
- *Server* menggunakan sistem operasi Ubuntu 16.04 dengan kapasitas 60 GB
- 1 *Domain* sebagai alamat yang diakses oleh perangkat IoT untuk melakukan komunikasi data. *Domain* yang melakukan penggantian IP jika terjadi *flooding*
- Beberapa alamat IP *public* untuk *server*, jika terjadi *flooding* agar bisa ganti ke IP lainnya.
- *Relational Database* MySQL.
- Pemrograman *server-side* PHP dengan *framework* Codeigniter untuk Restful API sebagai perantara antara *database* dengan perangkat IoT.

Berikut adalah desain struktur *database* IoT :

Column	Datatype	PK	NN
IoT_SN	INT(11)	☑	☑
IoT_type	CHAR(1)	☐	☑
IoT_productname	VARCHAR(45)	☐	☑
IoT_latitude	DECIMAL(9,6)	☐	☑
IoT_longitude	DECIMAL(9,6)	☐	☑
IoT_activeminutes	INT(11)	☐	☑
IoT_powerconsumption	INT(11)	☐	☑
IoT_modifieddate	TIMESTAMP	☐	☑

Gambar 3. Struktur *Database* IoT

Setelah proses produksi perangkat IoT dilakukan pendaftaran perangkat ke *database* langsung dengan *serial number*, tipe produk dan nama produknya. Proses pendaftaran perangkat IoT juga bisa melalui aplikasi dengan pengembangan lebih lanjut dari penelitian ini agar pengguna mendapatkan *user interface* dan *user experience* yang lebih baik.

Perangkat IoT yang sudah dipasarkan akan berkomunikasi dengan *server cloud* dengan mengirimkan data setiap perangkat IoT dinyalakan. Data yang dikirimkan berupa *update* dari lokasi perangkat (*Latitude* dan *Longitude*), durasi aktif perangkat dan konsumsi energi dari perangkat. Sehingga data terus diperbarui secara berkala. Data dari luar perangkat IoT atau data yang tidak terdaftar akan diabaikan sehingga lalu lintas data ke *server cloud* menjadi lancar dan penyimpanan *database* tidak penuh. Semua hanya data dari perangkat IoT yang terdaftar saja.

Proses *update* data hanya bisa dilakukan melalui Restful API dan tidak bisa dilakukan melalui *database* langsung, untuk Restful API menggunakan pemrograman *server-side* PHP dengan *framework* Codeigniter [19], PHP adalah bahasa pemrograman *server side scripting* yang berkomunikasi dengan *database*. *Script* PHP akan diproses dan dijalankan oleh *server* saat ada *request* dari *client*. Jenis *server* yang sering digunakan bersama dengan PHP adalah web *server* antara lain Apache, Nginx, dan LiteSpeed. [20], [21]

Pada penelitian ini digunakan *library* Restful API untuk Codeigniter sehingga *controller* bisa langsung berkomunikasi dengan *database* tanpa melalui *model* dan *output* data langsung berupa JSON (*JavaScript Object Notation*). [22]

Berikut adalah *script* PHP Codeigniter dengan menggunakan *library* Restful API untuk melakukan *update* data ke *database* SQL melalui Restful API :

```php
<?php

defined('BASEPATH') OR exit('No direct script access allowed');

require APPPATH . '/libraries/REST_Controller.php';
use Restserver\Libraries\REST_Controller;

class IoT extends REST_Controller {

    function __construct($config = 'rest') {
        parent::__construct($config);
        $this->load->database();
    }

    function index_PUT() {
```

```
        $IoT_SN = $this->PUT('IoT_SN');
        $data = array(
            'IoT_Latitude' => $this->PUT('IoT_Latitude'),
            'IoT_Longitude' => $this->PUT('IoT_Longitude'),
            'IoT_activeminutes' => $this->PUT('IoT_activeminutes'),
            'IoT_powerconsumption' => $this->PUT('IoT_powerconsumption')
        );
        $this->db->where('IoT_SN', $IoT_SN);
        $update = $this->db->update('IoT', $data);
        if ($update) {
            $this->Response($data, 200);
        } else {
            $this->Response(array('status' => 'fail', 502));
        }
    }
}
?>
```

Metode Restful API yang digunakan untuk melakukan *update* data adalah metode *PUT*, sedangkan metode lainnya seperti *GET*, *DELETE* dan lainnya tidak digunakan karena Restful API hanya digunakan untuk melakukan *update* data perangkat IoT, tidak untuk membaca data IoT atau menghapusnya agar data tetap aman.

Domain iot.informate.id memiliki beberapa alamat IP sebagai cadangan, apabila terjadi gangguan pada IP utama seperti terkena *flooding* dan lainnya bisa diarahkan ke IP cadangan agar *database* IoT tetap bisa diakses.

Pengujian

Pengujian dilakukan dengan menggunakan metode *Black box Testing*. Berikut adalah hasil pengujian dengan metode *black box* pada penelitian ini :

Tabel 1. Hasil Pengujian *Black box*

Proses	Hasil yang Diharapkan	Hasil	Keterangan
Input data perangkat IoT setelah diproduksi (pendaftaran perangkat IoT)	Berhasil dengan Status: *Query Completed*	Status: 1 *row(s) affected*, 0.185 *sec*	*Valid*, proses pendaftaran perangkat IoT berhasil
Update data perangkat IoT melalui Restful API	Berhasil dengan Status: 200 OK	Status: 200 OK, *Time*: 94*ms*	*Valid*, proses *update* data IoT berhasil
Fail-over apabila IP utama tidak bisa diakses (pergantian IP)	Berhasil dengan Status: 200 OK	Status: 200 OK, *Time*: 133*ms*	*Valid*, proses *update* data IoT setalah pergantian IP berhasil
Komunikasi langsung dengan *database* melalui *port* 3306	Tidak terhubung	Tidak terhubung	*Valid*, *database* tidak bisa diakses secara langsung hanya bisa melalui API
Serangan *brute force* ke *database*	Gagal	Gagal, akses ke *database* ditutup	*Valid*, *database* tidak terkena serangan *brute force*

Berikut adalah proses pengujian yang dilakukan menggunakan metode *black box* :

1. *Input* data perangkat IoT setelah diproduksi (pendaftaran perangkat IoT)

Pendaftaran perangkat ke *database* dengan *serial number*, tipe produk dan nama produknya, *syntax query* SQL seperti berikut :

INSERT INTO `IoT` (`IoT_SN`, `IoT_type`, `IoT_productname`) VALUES ('123456', 'L', 'Smart Lamp');

Maka data produk IoT akan terdaftar seperti berikut :

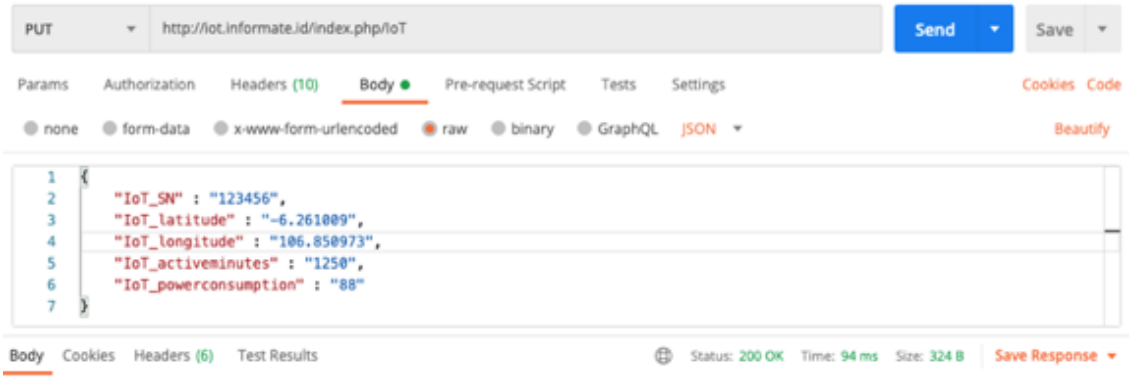
Gambar 4. Pendaftaran Perangkat IoT

2. *Update* data perangkat IoT melalui Restful API

Untuk *update* data, perangkat IoT mengakses alamat *http://iot.informate.id/index.php/IoT* dengan mengirim data dengan struktur JSON berikut :

```
{
   "IoT_SN" : "123456",
   "IoT_Latitude" : "-6.261009",
   "IoT_Longitude" : "106.850973",
   "IoT_activeminutes" : "1250",
   "IoT_powerconsumption" : "88"
}
```

Data pada *database* akan diperbarui sesuai dengan data yang dikirimkan dengan perantara Restful API. Berikut adalah contoh *update* data dengan menggunakan aplikasi *Postman* :



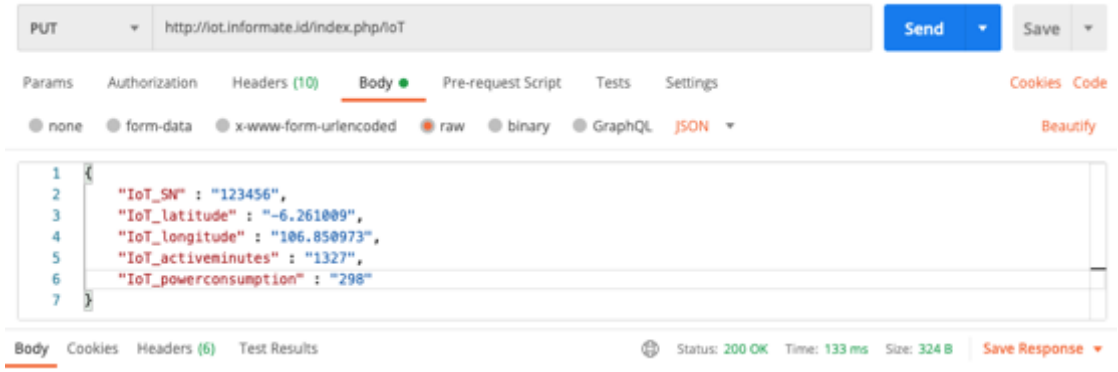
Gambar 5. *Update* Data IoT

3. *Fail-over* apabila IP utama tidak bisa diakses (pergantian IP)

Domain iot.informate.id memiliki beberapa alamat IP berikut sebagai cadangan :

Gambar 6. Daftar IP Cadangan (disamarkan)

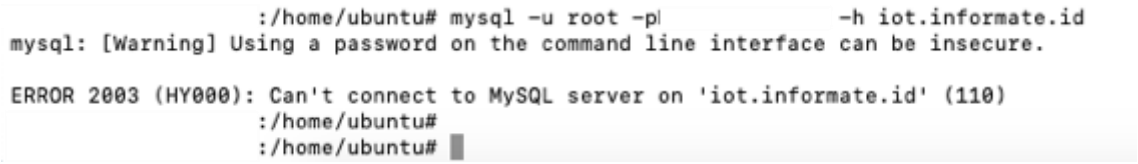
Setelah IP dirubah dilakukan uji coba *update* data melalui Restful API dengan menggunakan aplikasi *Postman* seperti berikut :



Gambar 7. *Update* Data IoT setelah Perubahan IP

4. Komunikasi langsung dengan *database* melalui *port* 3306

Database IoT hanya bisa diakses melalui Restful API dan akses langsung ke *database* melalui *port* 3306 (*port default* MySQL) ditutup agar *database* aman dari serangan. Ujicoba dilakukan melalui *client* dengan sistem operasi Ubuntu. Berikut adalah hasil uji coba akses *database* langsung melalui *port* 3306 :



Gambar 8. Uji Coba Akses *Database* IoT

5. Serangan *brute force* ke *database*

Uji coba serangan *brute force* menggunakan Hydra di sistem operasi Ubuntu. Serangan *brute force* menggunakan kombinasi 6 *username* dan 1000 *password* dengan total 6000 percobaan untuk masuk ke dalam *database* IoT. Berikut adalah hasil uji coba serangan *brute force* ke *database* :

```
                :/home/ubuntu# hydra -L username.txt -P password.txt iot.informate.id mysql
Hydra v8.1 (c) 2014 by van Hauser/THC - Please do not use in military or secret service organizations, or for illegal purposes.

Hydra (http://www.thc.org/thc-hydra) starting at 2021-03-19 18:07:24
[INFO] Reduced number of tasks to 4 (mysql does not like many parallel connections)
[WARNING] Restorefile (./hydra.restore) from a previous session found, to prevent overwriting, you have 10 seconds to abort...
[DATA] max 4 tasks per 1 server, overall 64 tasks, 6000 login tries (l:6/p:1000), ~23 tries per task
[DATA] attacking service mysql on port 3306
[STATUS] 6.00 tries/min, 6 tries in 00:01h, 5994 todo in 16:40h, 4 active
[STATUS] 7.33 tries/min, 22 tries in 00:03h, 5978 todo in 13:36h, 4 active
[STATUS] 7.71 tries/min, 54 tries in 00:07h, 5946 todo in 12:51h, 4 active
[STATUS] 7.60 tries/min, 114 tries in 00:15h, 5886 todo in 12:55h, 4 active
[STATUS] 7.55 tries/min, 234 tries in 00:31h, 5766 todo in 12:44h, 4 active


^Z
[4]+  Stopped                 hydra -L username.txt -P password.txt iot.informate.id mysql
```

Gambar 9. Serangan *Brute force* ke *Database* IoT

Dalam uji coba serangan *brute force* gagal terhubung ke *Database* IoT untuk mencoba 6000 kombinasi *username* dan *password* untuk memasuki *database*. Setelah 31 menit uji coba hanya terjadi 234 percobaan dengan 4 koneksi bersamaan dengan kecepatan masing - masing koneksi hanya 7 percobaan per menit dan semua digagalkan oleh *database* IoT.

Sebagai perbandingan jika serangan *brute force* berhasil 6000 percobaan bisa dilakukan dalam waktu 1 menit dengan 4 koneksi bersamaan dengan kecepatan masing – masing koneksi 3318 percobaan per menit. Berikut adalah contoh jika serangan *brute force* berhasil :

```
                :/home/ubuntu# hydra -L username.txt -P password.txt localhost mysql
Hydra v8.1 (c) 2014 by van Hauser/THC - Please do not use in military or secret service organizations, or for illegal purposes.

Hydra (http://www.thc.org/thc-hydra) starting at 2021-03-19 17:43:10
[INFO] Reduced number of tasks to 4 (mysql does not like many parallel connections)
[WARNING] Restorefile (./hydra.restore) from a previous session found, to prevent overwriting, you have 10 seconds to abort...
[DATA] max 4 tasks per 1 server, overall 64 tasks, 6000 login tries (l:6/p:1000), ~23 tries per task
[DATA] attacking service mysql on port 3306
[STATUS] 3318.00 tries/min, 3318 tries in 00:01h, 2682 todo in 00:01h, 4 active
1 of 1 target completed, 0 valid passwords found
Hydra (http://www.thc.org/thc-hydra) finished at 2021-03-19 17:45:06
```

Gambar 10. Contoh Perbandingan Serangan *Brute force*

4. KESIMPULAN DAN SARAN

Dari penelitian yang telah dilakukan, *Database* IoT berbasis *cloud* dengan Restful API lebih aman digunakan jika dibandingkan dengan *database* langsung bisa diakses oleh perangkat IoT atau internet. Peran Restful API sebagai *firewall* dan akses ke *database* secara langsung ditutup terbukti membuat *database* terhindar dari serangan *brute force* dari uji coba yang dilakukan.

Selanjutnya bisa dilakukan penelitian mengenai pengolahan data IoT yang telah tersimpan tersebut baik dengan metode *data mining*, atau integrasi dengan aplikasi *business intelegence*, atau aplikasi sebagai *user interface* dari data tersebut hingga integrasi dengan *Google Maps*.

DAFTAR PUSTAKA

[1] A. Prasetyo and A. R. Yusuf, "Integrated Device Electronic Untuk Sistem Irigasi Tetes Dengan Kendali Internet of Things," *J. Ilm. Teknol. Inf. Asia*, vol. 14, no. 1, p. 1, 2019, doi: 10.32815/jitika.v14i1.361.

[2] K. Dewi and S. Sulaeman, "Penerapan Teknologi Integrated Device Electronic (Ide) Untuk Peningkatan Produktifitas Hasil Pertanian Pada Purwarupa Kumbung Jamur Tiram Di Dataran Rendah," *Semin. Nas. Has. Penelit. (SNP2M PNUP)*, vol. 0, no. 0, pp. 154–159, 2018, [Online]. Available: http://jurnal.poliupg.ac.id/index.php/snp/article/view/784/0.

[3] M. Luckies, G. Azis, E. S. Pramukantoro, and R. A. Siregar, "Pengembangan Internet Gateway Device berbasis Koneksi GPRS untuk Mengoleksi dan Meneruskan Data ke Media Penyimpanan," *J. Pengemb. Teknol. Inf. dan Ilmu Komput. Univ. Brawijaya*, vol.

3, no. 1, pp. 40–46, 2019.

[4] E. F. Kurniawan, E. S. Pramukantoro, and F. A. Bakhtiar, "Implementasi Perangkat Internet Gateway Device Untuk Menghubungkan Infrastruktur IoT dan Aplikasi Cloud Menggunakan Narrowband Internet of Things (NB-IoT)," *J. Pengemb. Teknol. Inf. dan Ilmu Komput.*, vol. 3, no. 6, 2019.

[5] N. Fath and R. Ardiansyah, "Sistem Monitoring Alat Pemberi Pakan Ikan Otomatis Menggunakan NodeMCU Berbasis Internet of Things," *Techno.Com*, vol. 19, no. 4, pp. 449–458, 2020, doi: 10.33633/tc.v19i4.4051.

[6] C. Stergiou, K. E. Psannis, B. G. Kim, and B. Gupta, "Secure integration of IoT and Cloud Computing," *Futur. Gener. Comput. Syst.*, vol. 78, pp. 964–975, 2018, doi: 10.1016/j.future.2016.11.031.

[7] A. Munir, P. Kansakar, and S. U. Khan, "IFCIoT: Integrated Fog Cloud IoT: A novel architectural paradigm for the future Internet of Things," *IEEE Consum. Electron. Mag.*, vol. 6, no. 3, pp. 74–82, 2017, doi: 10.1109/MCE.2017.2684981.

[8] G. Neagu, S. Preda, A. Stanciu, and V. Florian, "A Cloud-IoT based sensing service for health monitoring," *2017 E-Health Bioeng. Conf. EHB 2017*, pp. 53–56, 2017, doi: 10.1109/EHB.2017.7995359.

[9] M. K. Anwar and D. Firdaus, "IMPLEMENTATION OF FACE SIMILARITY USING TINY FACE DETECTOR," *Int. J. Inf. Syst. Comput. Sci.*, vol. 4, no. 1, pp. 22–28, 2020.

[10] P. Setialana, T. B. Adji, and I. Ardiyanto, "Perbandingan Performa Relational, Document-Oriented dan Graph Database Pada Struktur Data Directed Acyclic Graph," *J. Buana Inform.*, vol. 8, no. 2, pp. 77–86, 2017, doi: 10.24002/jbi.v8i2.1079.

[11] A. Budiman and Noviardi, "Informasi Penerapan Keamanan Penggunaan Data Pada Database Kepegawaian Menggunakan Teknik Transparent Data Encryption," *SATIN-Sains dan Teknol. Inf.*, vol. 2, no. 2, 2016.

[12] A. Kusyanti2, K. Amron, and F. Mohammad, "Pengamanan Data pada Media Penyimpanan Cloud Menggunakan Teknik Enkripsi dan Secret Sharing," *J. Pengemb. Teknol. Inf. dan Ilmu Komput.*, vol. Vol. 2, No. 11,u, no. 11, pp. 4863–4869, 2018, [Online]. Available: http://j-ptiik.ub.ac.id.

[13] X. Chen, Z. Ji, Y. Fan, and Y. Zhan, "Restful API Architecture Based on Laravel Framework," *J. Phys. Conf. Ser.*, vol. 910, no. 1, 2017, doi: 10.1088/1742-6596/910/1/012016.

[14] E. Susanti, "Implementasi RESTful API dalam Pembuatan Master Data Planogram Menggunakan Framework Flask (Studi Kasus: PT Sumber Alfaria Trijaya, Tbk)," *Techno.Com*, vol. 19, no. 3, pp. 295–307, 2020, doi: 10.33633/tc.v19i3.3468.

[15] A. S. Mustaqim, D. Kurnianto, and F. T. Syifa, "Implementasi Teknologi Internet of Things Pada Sistem Pemantauan Kebocoran Gas LPG dan Kebakaran Menggunakan Database Pada Google Firebase," *Elektron J. Ilm.*, vol. 12, no. 1, pp. 34–40, 2020, doi: 10.30630/eji.12.1.161.

[16] D. M. Ibrahim, R. Primananda, and M. Data, "Perbandingan Performa Database Apache HBase dan Apache Cassandra Sebagai Media Penyimpanan Data Sensor Internet of Things," *Pengemb. Teknol. Inf. dan Ilmu Komput.*, vol. 2, no. 8, pp. 2943–2949, 2018.

[17] E. Edy, F. Ferdiansyah, W. Pramusinto, and S. Waluyo, "Pengamanan Restful API menggunakan JWT untuk Aplikasi Sales Order," *J. RESTI (Rekayasa Sist. dan Teknol. Informasi)*, vol. 3, no. 2, pp. 106–112, 2019, doi: 10.29207/resti.v3i2.860.

[18] M. F. A. Tansa Trisna Astono Putri, Mhd. Dominique Mendoza, "Bruteforce In The Hydra Process And Telnet Service Using The Naïve Bayes Method," *J. Mantik*, vol. 3, no. January, pp. 31–38, 2019.

[19] A. Subari, D. Y. Tadeus, H. Winarno, and T. Yuwono, "Rancang Bangun Sistem Administrasi Kerja Praktek Dan Tugas Akhir Berbasis Web Menggunakan Framework Codeigniter," *Gema Teknol.*, vol. 19, no. 4, p. 1, 2018, doi: 10.14710/gt.v19i4.19147.

[20] Y. W. Luthfi Muhammad, Data Mahendra, "Perbandingan performa reverse proxy caching nginx dan varnish pada web server apache," *Pengemb. Teknol. Inf. dan Ilmu*

Komput., vol. 2, no. 4, pp. 1457–1463, 2018.

[21] A. Y. Chandra, "Analisis Performansi Antara Apache & Nginx Web Server Dalam Menangani Client Request," *J. Sist. dan Inform.*, vol. 14, no. 1, pp. 48–56, 2019, doi: 10.30864/jsi.v14i1.248.

[22] B. W. PUTRA, A. SAPUTRA, M. R. SANJAYA, and D. KURNIAWAN, "Enabling Collaboration of CodeIgniter Framework and RESTful API for Utilize Web Mobile Interface Implemented on Final Project Management System," *Adv. Intell. Syst. Res.*, vol. 172, 2020, doi: 10.2991/aisr.k.200424.080.