Pengembangan teknologi *Medical Drone* Serta Analisis Aspek Sekuriti dan Vulnerabilitas Dalam Penggunaannya

Aaron Christopher Tanhar (07211940000055)
Departemen Teknik Komputer
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia 60111
christopher.19072@mhs.its.ac.id

***Abstrak*—Integritas dari sebuah File Digital merupakan salah satu aspek penting dari sekuriti sistem komputer. Terdapat instilah yang dinamakan dengan File Integrity Monitoring (FIM) yang merupakan sebuah proses yang melakukan tindakan validasi dari sebuah File pada Operating System dan software aplikasi menggunakan metode verifikasi antara state atau keadaan yang terkini dengan state yang diketahui atau sebelumnya. Tentu saja hal tersebut penting untuk dilakukan sehingga file-file yang selalu kita transmisikan baik melalui lokal komputer dan melalui nirkabel akan selalu terjamin isinya sehingga tidak korup. Apabila tidak demikian maka mungkin dapat terjadi kerusakan atau pemalsuan data. Maka pada makalah ini saya melakukan riset terhadap metode yang dapat menjamin integritas pada file digital di sistem komputer. Dari segi security, terdapat beberapa penelitian yang juga melibatkan perkara *file integrity*. Penelitian dengan tema secure messaging juga memberikan kita penemuan yang relevan tentang usability dan security dari proses autentikasi pengguna layanan messaging tersebut. Parity Bit atau biasa juga disebut dengan *Check Bit* adalah bentuk sederhana dari *error detecting code*. Terdapat dua varian dari bit paritas, paritas genap dan paritas ganjil. Sebuah checksum merupakan suatu blok data berukuran kecil yang diperoleh dari blok data digital yang lainnya. Sebuah *MD5 message-digest algorithm* adalah hash function yang sering digunakan untuk mengecek integritas pada file. MD5 digest sudah digunakan secara luas pada dunia perangkat lunak untuk memberikan sebuah jaminan dimana file yang ditransmisikan telah tiba dan datanya sama dengan data yang asli. SHA-1 merupakan fungsi *hash cryptographic* yang menerima input lalu akan mengeluarkan output 160-bit (20 byte) nilai hash yang dikenal sebagai *message digest*. SHA-2 merupakan perubahan yang cukup signifikan dibandingkan pendahulunya, SHA-1. SHA-3 adalah subset dari Keccak yang didasari dari pendekatan baru yang dinamakan sponge construction. Sponge construction didasari dari fungsi random atau fungsi permutasi. Sebuah *Cyclic Redundancy Check* adalah *error-detecting code* yang biasa digunakan pada jaringan digital dan storage untuk mendeteksi adanya perubahan yang tidak diinginkan pada data. Komputasi dari CRC diturunkan dari polynomial division, modulo dua.**

***Kata kunci*—*Integrity*, *Security*, Komputer, File, *MD5*, *Checksum*, *SHA*, *Digest*, *CRC*.**

I. Pendahuluan

Drone atau yang biasa juga disebut sebagai pesawat nirawak merupakan pesawat yang tidak memiliki pilot manusia, kru, maupun penumpang yang membuat pesawat ini sepenuhnya independen. Pesawat nirawak yang juga disebut dalam bahasa inggris sebagai *UAV* atau *Unmanned Aerial Vehicle* merupakan sebuah komponen daripada *UAS* atau *Unmanned Aerial System* yang menyertakan kontroller yang berbasis di daratan dan sistem komunikasi dengan *UAV*-nya itu sendiri [1]. Terbangnya pesawat nirawak dapat dikendalikan oleh manusia sebagaimana *remote-piloted aircraft(RPA)*, atau dengan beberapa teknik autonomous, seperti bantuan autopilot, hingga pesawat yang benar-benar autonomous sehingga sama sekali tidak ada intervensi dari manusia [2].

UAV ini awalnya dikembangkan selama abad ke-20 yang diutamakan untuk melakukan misi militer yang dianggap terlalu kotor atau berbahaya untuk dilakukan oleh manusia, dan pada abad ke-21 drone ini sudah menjadi hal krusial yang harus ada pada kemiliteran. Seiring dengan berjalannya waktu dan harga dari drone atau UAV ini menurun maka penggunaannya merambah ke hal-hal yang tidak berbau militer [3][4]. Hal ini termasuk dengan fotografi udara, pengantaran produk atau barang, agrikultur, surveillance, inspeksi infrastruktur, sains[5][6][7][8], penyelundupan[9], dan balapan drone.

Teknologi drone menyediakan keuntungan yang sangat melimpah dan memberi kesempatan yang luas untuk banyak bidang penelitian. Drone dapat melakukan hal-hal seperti halnya surveying, humanitarian work, manajemen resiko bencana, riset dan juga transportasi[10]. Dalam bidang agrikultur, drone dapat melakukan imagery real-time dan sensor data dari lahan pertanian yang luas yang tidak dapat diakses dengan cepat menggunakan kaki ataupun kendaraan[10].

Global Positioning System atau GPS dan juga aplikasi untuk smartphone dan tablet dan meningkatkan kualitas prediksi durasi penerbangan, lebih reliable, dan kemudahan penggunaan serta kemampuan untuk memanfaatkan kamera yang lebih baik dan juga sensor-sensor lain yang dibutuhkan untuk mengaplikasikan drone pada agrikultur dan sumber daya alam[10]. Penggunaan drone menjadi sangat intim dengan beberapa sektor yang dikembangkan dengan ekonomi berkembang. Jika kita kehilangan drone itu maka dapat mengakibatkan implikasi yang berakibat merusak.

Penggunaan drone yang kerap terjadi pada bidang kesehatan atau medis biasanya berupa penyaluran alat-alat paket pertolongan pertama, obat-obatan, penyaluran vaksin, darah, dan kebutuhan kesehatan lainnya yang ditujukan ke daerah terpencil. Hal ini dapat memberikan transportasi test sample yang aman dari penyakit dengan tingkat penularan yang tinggi karena tidak memerlukan manusia untuk langsung terjun di lapangan, dan juga dapat memberikan akses cepat kepada external defibrilator otomatis untuk pasien yang menderita *cardiac arrest* untuk menyelamatkan nyawanya[10] dan selama masa gawat darurat kesehatan. Di era pandemi COVID-19 ini drone dapat mengantar atau menyalurkan *Personal Protective Equipments* (PPE), alat test, vaksin, pengobatan, dan sample dari laboratorium.

Drone dapat membantu untuk melakukan inspeksi social distancing yang mudah secara otomatis[11]. Sebagai teknologi baru, drone dapat menyediakan solusi dari konteks pada keadaan ekstrim darurat, topografi yang sulit, dan infrastruktur transportasi. Pengadopsian drone untuk melakukan pengantaran atau penyaluran benda-benda yang krusial dan obat-obatan yang penting untuk keselamatan kepada seluruh masyarakat dengan keadaan ekonomi apapun dapat mewujudkan kesetaraan kesehatan universal[12]. Keuntungan yang didapatkan dari drone pada sektor transportasi adalah keuntungan logistik dan juga transportasi penumpang[13].

Ekspansi dari penggunaan drone yang pada awalnya hanya digunakan untuk keperluan militer hingga menjadi keperluan sipil juga membuat adanya urgensi untuk melakukan pengembangan teknologi dari drone itu sendiri menjadi lebih baik dan memanfaatkan segala potensi drone yang ada demi masyarakat yang lebih baik. [14, Greenwood (2016)] juga mengatakan bahwasanya untuk menyadari potensi penuh dari sebuah teknologi drone ini, peraturan yang meregulasi penggunaan drone ini sangat diperlukan sembari tetap mengutamakan keamanan masyarakat dan hak-hak privasinya juga. Penyalahgunaan seperti contohnya terorisme, privasi dan penggunaan militer merupakan resiko yang dikhawatirkan terjadi pada penggunaan drone[15].

Meskipun terdapat isu-isu tersebut, [16, Sylvester (2018)] mengatakan bahwa teknologi drone ini dapat memberikan pekerjaan untuk para pemuda yang dapat menggunakan drone untuk menyediakan layanan untuk petani-petani di daerah pedesaan. Berlawanan dengan latar belakang ini, review ini berusaha untuk memperlihatkan pengembangan saat ini daripada penggunaan drone yang ada di sektor agrikultur, bidang kesehatan, dan juga kemiliteran. Untuk menggapai target ini, kita awalnya harus diperlihatkan latar belakang teknikal secara singkat dari drone ini dan pada sektor ini dan juga reviewnya mengambil dari pendekatan analisis SWOT.

Internet of Drones (IoD) meniru akronim dari IoT yang menempatkan "Drones" untuk menggantikan "Things". Maka, IoD ini memiliki beberapa kesamaan dengan IoT atau Internet of Things ini. Gharibi et al. [17] mendefinisikan IoD sebagai arsitektur jaringan kontrol yang berlapis yang dapat membantu untuk mengkoordinasikan drone-drone [18]. Paradigma jaringan IoD ini dapat diaplikasikan dalam operasi Search and Rescue, monitoring angkatan militer, inspeksi industrial, monitoring infrastruktur, sistem pengantaran barang[19], agrikultur [20] [21], mapping supply chain, manajemen bencana [22] [23], dan lain sebagainya. Terdapat ekspektasi kuat yang adalah IoD dapat memiliki peran yang signifikan pada smart city di masa depan [24]. Layanan publik tingkat lanjut sekarang biasanya sekarang dapat mengadakan operasi risiko kritikal alami maupun buatan manusia dengan menggunakan IoD [25] [26].

Akan tetapi, jaringan IoD ini dapat menjadi target dari beberapa ancaman keamanan dan privasi yang berbahaya dan jahat. Baik drone-drone maupun entiti IoD lain mungkin saja dibajak untuk tujuan serangan siber, data breaches, atau pencurian data dengan menggunakan payload. Berdasarkan dari author pada [27], sebuah drone phantom DJI ketika dibajak dapat dijual belikan pada situs ebay dengan harga sebesar 1,000 US dollar. Authornya juga menyatakan bahwa sebuah kamera drone yang digunakan pada industri film dapat dihargai sampai dengan harga 20,000 US dollar, dan sebuah detektor cahaya dan range (LIDAR) sensor dapat diberi harga hingga mencapai 50,000 US dollar. Lebih lagi, ketika sebuah drone yang membawa data yang berharga dibajak, kerugiannya dapat mencapai ribuan US dollar. Kerugian yang jauh lebih besar dapat terjadi ketika drone yang diserang merupakan drone untuk militer. Dampaknya bukan saja hanya membocorkan data berharga atau rahasia ataupun kerusakan fisik dari drone bersangkutan namun juga drone yang dibajak dapat dijadikan sebagai sebuah senjata oleh orang yang membajak [27]. Komunikasi yang terjadi antara drone yang ada di dalam jaringan IoD adalah melalui internet yang tidak aman (umumnya jaringan nirkabel atau wireless maupun WiFi) dan menggunakan sinyal navigasi (contohnya global positioning system (GPS)) [28].

Hal ini dapat mempengaruhi aspek privasi dan keamanan pada drone secara signifikan. Hacker yang tidak bertanggungjawab dapat dengan mudah mengakses konfigurasi dari drone dan membajaknya dengan menggunakan aplikasi open-source untuk membajak drone (contohnya skyjack) dan secara nirkabel mendapatkan kendali dari drone yang menjadi target. Kebanyakan ancaman privasi dan keamanan yang ada pada drone sipil terjadi karena kesalahan pada desainnya. Kebanyakan drone dirancang tanpa perlindungan internet security dan mekanisme autentikasi [29]. Meskipun secara tingkat kesulitan lebih sulit untuk membajak drone pada militer dikarenakan infrastruktur keamanannya yang lebih tinggi apabila dibandingkan dengan drone sipil, seorang hacker yang handal dapat menggunakan teknik yang lebih canggih. Sebuah contoh dari hal ini adalah CIA RQ-170 Sentinel US spy drone dibajak oleh hacker yang berasal dari Iran pada desember 2011 [30].

Sudah banyak teknik sekuriti, keamanan dan privasi yang dikembangkan oleh peneliti-peneliti untuk mendapatkan jaminan dari keamanan dari jaringan *IoD* atau *Internet of Drones* ini. Teknik yang ditujukan kepada memitigasi atau mencegah masalah yang mempengaruhi lokalisasi keamanan dari drone atau kebutuhab sekuriti yang diasosiasikan dengan jaringan IoD. Serangan lokalisasi error mengganggu kemampuan positioning daripada drone yang terdapat di dalam jaringan IoD sehingga

dapat menyebabkan kerusakan yang besar pada performa keseluruhan dari jaringan IoD. Lebih lagi, kebutuhan privasi dan keamanan merupakan tujuan yang menentukan kesanggupan dan fungsi dari jaringan IoD yang didapatkan dari memitigasi ancaman keamanan dan privasi tertentu [31]. Kebutuhan akan keamanan dan privasi dari jaringan IoD ini termasuk dengan integrity, availability, confidentiality, dan privacy preservation.

Pembahasan pada paper ini dimulai dengan presentasi mengenai penelitian lain (Bagian II). Kemudian dilanjutkan dengan penjelasan mengenai hal-hal apa saja yang menarik dari parity bit (Bagian III). Setelah itu dilanjutkan dengan pemahaman dari istilah checksum (Bagian IV). Pada bagian tersebut juga diklasifikasikan menjadi beberapa sub bagian seperti MD5, SHA-1, SHA-2 dan SHA-3. Dengan didasari bagian-bagian sebelumnya, maka dilanjutkan dengan Cyclic Redundancy Check (Bagian V) Terakhir, didapatkan kesimpulan dari penelitian yang telah dilakukan (Bagian VI).

II. Penelitian Terkait

Aspek keamanan dalam drone sipil sudah direview pada [32]. Kemudian, beberapa serangan keamanan pada drone sudah dianalisis pada [33], [34], [35], [36], dan [37]. Metode-metode untuk melakukan deteksi drone dianalisis pada [38], [39], [40], dan [41]. Akan tetapi, keterbatasan utama dari penelitian-penelitian sebelumnya adalah kurangnya analisis yang lebih mendalam dari vulnerabilities dari dronenya itu sendiri serta kurangnya analisis attack life cycle. Terlebih lagi, hanya satu aspek dari keamanan drone yang dianalisis, yakni penyerangan pada drone. Teknik penanggulangan yang saat ini telah ada perlu dianalisis, dan teknik baru perlu diusulkan untuk mengatasi kekurangan dari solusi keamanan pada drone yang ada pada saat ini.

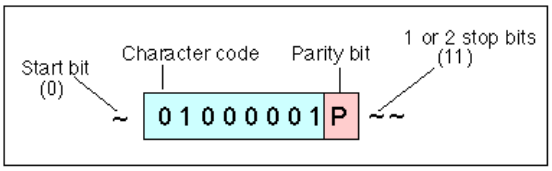
Menurut Yao and Ansari [42], arsitektur IoD pertama dirancang oleh Gharibi et al. [17]. Arsitektur tersebut terdiri dari lima layer konseptual (air space layer, node-to-node layer, end-to-end layer, services layer, dan application layer). Setiap layer dapat mengakses layanan yang sudah diberikan oleh layer dibawah layer tersebut. Lin et al. [43] melakukan penelitian lebih lanjut tentang arsitektur milik gharibi dan menunjukkan kelebihan dan kelemahan dari arsitektur tersebut. Arsitektur tersebut dapat memberikan pencegahan terhadap tabrakan drone saat berada di udara. Kemudian juga dapat memberikan kontrol lebih dimana tempat yang dapat dicapai dan tidak oleh drone. Akan tetapi, terdapat kelemahan yang mana adalah kurangnya penjaluran efektif, kontrol penyumbatan, dan tantangan akan keamanan dan privasi (penyaluran data yang tidak aman). Author mengusulkan solusi yang kira-kira dapat mengatasi permasalahan yang sudah dianalisis yang nantinya akan sesuai dengan arsitektur jaringan IoD.

Terlebih lagi, author di [44] mengusulkan penambahan teknologi blockchain pada layer IoD untuk membuat IoD semakin rahasia, aman, dan *tamperproof*. Qureshi et al. [45] mengusulkan sebuah arsitektur IoD berbasis cloud untuk menyediakan virtualisasi akses pada drone melalui cloud dan mengunggah komputasi yang berat ke cloud dengan batasan resource yang terbatas. Arsitektur ini tersusun atas tiga buah layer. Layer yang pertama adalah drone layer yang merepresentasikan susunan resource ataupun layanan yang diberikan kepada end-users. Pada layer kedua, layer ini disebut sebagai layer layanan cloud. Layer kedua terdiri atas tiga komponen (komponen penyimpanan yang berguna untuk menyimpan data yang didapatkan dari drone, komponen komputasi, dan komponen interface atau antarmuka). Akhirnya, layer ketiga disebut sebagai layer klien. Layer ini memiliki antarmuka dari kedua layer sebelumnya yaitu layer drone dan layer cloud.

III. Parity Bit

Pada ilmu Matematika, paritas merupakan sebuah properti dari sebuah *integer* yang menentukan apabila *integer* tersebut ganjil atau genap. Sebuah paritas dari bilangan bulat adalah genap apabila jika bilangan tersebut dapat dibagi dengan dua dan sisanya adalah nol. Sebaliknya jika sisanya satu maka paritasnya adalah ganjil. *Parity Bit* atau biasa juga disebut dengan *Check Bit* adalah bentuk sederhana dari *error detecting code*. Error detecting code ini biasa digunakan untuk melakukan transmisi digital data yang *reliable* dengan menggunakan kanal komunikasi yang *unreliable*. Semua bentuk error detection akan menambahkan beberapa data ekstra pada pesan, yang dapat digunakan oleh penerima untuk melakukan verifikasi dari konsitensi pada pesan tersebut. *Parity bit* akan memastikan bahwa jumlah total dari 1-bit pada sebuah string adalah ganjil atau genap [46].

Terdapat dua varian dari bit paritas, paritas genap dan paritas ganjil. Pada kasus paritas genap, untuk setiap bit dari string yang diberikan, jumlah bit yang bernilai 1 akan dihitung. Apabila jumlahnya ganjil, maka nilai dari bit paritas akan diatur menjadi 1, yang membuat total bit 1 menjadi bernilai genap. Sebaliknya, jika jumlah dari bit 1 pada awalnya sudah genap, maka bit paritas akan diatur menjadi 0. Pada kasus paritas ganjil, prosedurnya akan berkebalikan. Untuk setiap bit dari string yang diberikan, jumlah bit yang bernilai 1 akan dihitung. Apabila jumlahnya genap, maka nilai dari bit paritas akan diatur menjadi 1, yang membuat total bit 1 menjadi bernilai ganjil. Sebaliknya, jika jumlah dari bit 1 pada awalnya sudah ganjil, maka bit paritas akan diatur menjadi 0.



Gambar 1. Struktur parity bit

Apabila sebuah string yang berisi bit-bit paritas ditransmisikan secara tidak benar atau korup, maka bit paritasnya akan salah. Hal tersebut mengindikasikan terjadinya *parity error* saat transmisi. Bit paritas ini hanya bisa mendeteksi adanya error; tidak dapat melakukan koreksi error pada file yang terjadi, karena tidak ada mekanisme yang memungkinkan bit paritas untuk mengetahui bit yang mana yang korup. Lalu apabila meninjau Tabel I, dapat diketahui bahwa bit paritas ini

Tabel I
CONTOH BIT PARITAS

7 bits of data	Count of 1 bits	8 bits including parity even	odd
0000000	0	00000000	00000001
1010001	3	10100011	10100010
1101001	4	11010010	11010011
1111111	7	11111111	11111110

memiliki kelemahan. Bit paritas hanya mengecek jumlah bit ganjil atau genap tanpa meninjau jumlah asli bitnya. Hal ini mengakibatkan terjadinya kegagalan dalam menangkap error pada data. Sebagai contoh, jika A ingin mengirim data `1001` dengan paritas genap maka bit paritasnya adalah `0` sehingga yang ditransmisikan ke B adalah `10010`. Terjadi error pada transmisi, sehingga yang sampai ke B adalah `11011`. Akan tetapi, B menghitung jumlah bit dan mendapati genap, sehingga B akan menganggap data tersebut benar walaupun sebenarnya salah.

```
def even_parity(x):
    parity = 0
    while x:
        parity ^= x & 1
        x >>= 1
    return parity
```

Listing 1. Program perhitungan bit paritas genap.

```
def odd_parity(x):
    parity = 1
    while x:
        parity ^= x & 1
        x >>= 1
    return parity
```

Listing 2. Program perhitungan bit paritas ganjil

Potongan kode pada Listing 1 diatas adalah contoh kode yang berfungsi menemukan bit paritas genap. Untuk bit paritas ganjil tinggal mengganti inisialisasi variabel parity menjadi 1. Inputnya adalah *integer* yang bisa kita jadikan biner dulu, seperti `0b10001`.

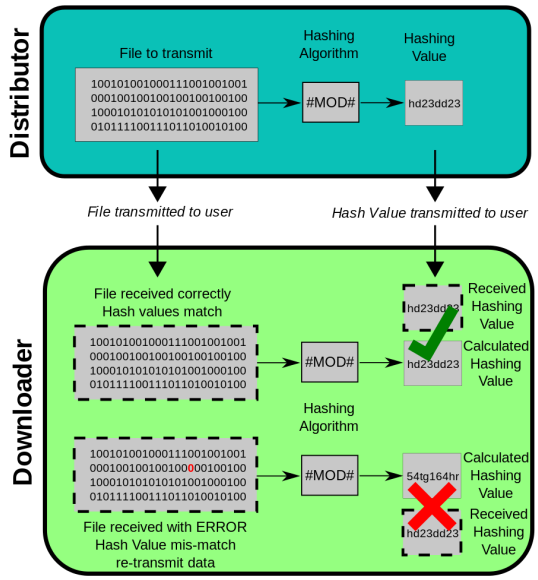
IV. CHECKSUM

Sebuah checksum merupakan suatu blok data berukuran kecil yang diperoleh dari blok data digital yang lainnya. Checksum memiliki tujuan untuk mendeteksi error yang mungkin saja terjadi ketika melakukan transmisi data pada memori maupun internet. Checksum biasanya digunakan untuk verifikasi *integrity* dari sebuah data, dengan kata lain apakah datanya sudah dimodifikasi setelah checksum dibuat. Output dari sebuah algoritma fungsi *hash cryptographic*, yang biasa disebut *hashes* atau *digest* digunakan sebagai checksum. Checksum biasanya direpresentasikan dengan menggunakan string hexadecimal (e.g., 69fac420b...), ukurannya berkisar antara 32 sampai 128 digit. Fungsi *hash cryptographic* ini memiliki tiga properti utama, yakni pre-image resistance, second pre-image resistance, dan collision resistance [47]. Fungsi *hash cryptographic* yang sering digunakan adalah MD5, SHA-1, dan SHA-2. MD5 merupakan salah satu fungsi *hash cryptographic* pertama yang dicanangkan. Namun pada akhir 1990an, algoritma tersebut mulai ditinggalkan karena berhasil dirusak sehingga tidak baik untuk keamanan. SHA-1 direkomendasikan oleh National Institute of Standards and Technology (NIST) sampai pada tahun 2015, algoritmanya terbobol. Sekarang, SHA-2 menjadi algoritma terpopuler yang direkomendasikan NIST untuk melakukan verifikasi integritas file [48].

A. MD5

Sebuah *MD5 message-digest algorithm* adalah hash function yang *cryptographically broken* tapi masih sering digunakan untuk membuat nilai hash 128 bit. *Cryptographically broken* maksudnya adalah secara security sudah tidak aman atau *vulnerable*. Pada awalnya, MD5 didesain sebagai fungsi *hash cryptographic*, namun ditemukan banyak vulnerability pada MD5, yang membuat algoritma ini ditinggalkan. MD5 masih bisa digunakan untuk melakukan verifikasi integritas data, namun hanya untuk mengecek kerusakan data yang tidak disengaja. MD5 masih cocok digunakan untuk hal-hal non kriptografik, seperti menentukan partisi dari kunci tertentu pada database yang sudah dipartisi, dan juga beberapa orang masih memilih menggunakan MD5 daripada *Secure Hash Algorithm* (SHA) karena beban komputasi yang lebih ringan [49]. MD5 didesain oleh Ronald Rivest tahun 1991 untuk menggantikan MD4, dan dispesifikasikan pada tahun 1992 sebagai RFC 1321. Salah satu kebutuhan dasar dari fungsi *hash cryptographic* adalah harus secara komputasional tidak mungkin ditemukan dua buah data yang berbeda dan memiliki nilai hash yang sama. MD5 gagal melakukan hal tersebut dan berdampak cukup fatal. Pada tahun 2019, MD5 tetaplah secara luas digunakan, meskipun kelemahan yang dimiliki sudah banyak didokumentasikan oleh para ahli security [50]. Sebuah *collision attack* dengan menggunakan komputer 2.6 GHz Pentium 4 dapat menemukan collision pada MD5 dalam hitungan detik. MD5 menggunakan *Merkle–Damgård construction*, jadi jika dua prefix dengan hash yang sama dapat dibuat, maka suffix umum dapat ditambahkan pada keduanya untuk membuat collision yang terjadi dapat diterima dan dianggap valid pada aplikasi yang menggunakannya.

MD5 digest sudah digunakan secara luas pada dunia perangkat lunak untuk memberikan sebuah jaminan dimana file yang ditransmisikan telah tiba dan datanya sama dengan data yang asli. Sebagai contoh, file pada server-server umumnya menyediakan MD5sum yang sudah dikomputasi untuk file tertentu, sehingga user yang mengunduh file tersebut dapat membandingkan apakah file yang diterima sama dan utuh dengan file asli yang terdapat pada server. Kebanyakan UNIX-based system menggunakan MD5sum pada package managernya. ROM pada Android juga menggunakan checksum jenis MD5 ini. MD5 juga biasa digunakan untuk hashing password satu arah, namun NIST tidak merekomendasikan MD5 untuk keperluan ini



Gambar 2. File transmisi dengan hashing

MD5 akan memproses pesan yang memiliki panjang bervariasi menjadi output dengan panjang 128 bit. Pesan masukan dipecah menjadi block-block 512 bit, dan pada pesan dilakukan padding agar panjangnya dapat dibagi dengan 512. Cara paddingnya adalah: pertama, satu bit, 1 ditambah pada akhir pesan. Kemudian bit ini diikuti dengan nol sebanyak yang dibutuhkan untuk membuat panjang message kurang dari 64 bit dari kelipatan 512. Bit sisa yang panjangnya 64 bit diisi dengan 64 bit yang merepresentasikan panjang pesan mulanya, modulo 2.

Algoritma inti dari MD5 bekerja pada keadaan 128-bit, dibagi menjadi 32-bit words, dinotasikan dengan A, B, C, dan D. Variabel tersebut diinisialisasi ke konstanta tertentu. Algoritma inti akan menggunakan tiap blok pesan berukuran 512-bit untuk memodifikasi statenya. Pemrosessan dari blok pesan terdiri dari 4 tahap, termed rounds; tiap tahap terdiri dari 16 operasi yang mirip berdasarkan fungsi linear F, penjumlahan modular, dan rotasi ke kiri. Berikut persamaannya. Terdapat 4 fungsi yang mungkin terjadi, fungsi berbeda digunakan untuk tiap tahapan atau round.

$$F(B, C, D) = (B \wedge C) \vee (\neg B \vee D) \quad (1)$$

$$G(B, C, D) = (B \wedge D) \vee (B \vee \neg D) \quad (2)$$

$$H(B, C, D) = B \oplus C \oplus D \quad (3)$$

$$I(B, C, D) = C \oplus (B \vee \neg D) \quad (4)$$

```
import hashlib

file_name = 'filename.exe'

original_md5 = '6941402abc4b2a76b9719d911420c592'

with open(file_name) as file_to_check:
    data = file_to_check.read().encode()
    md5_returned = hashlib.md5(data).hexdigest()

if original_md5 == md5_returned:
    print('MD5 verified.')
else:
    print('MD5 verification failed!.')
```

Listing 3. Program python sederhana untuk cek md5sum

Pada Listing 3 adalah contoh program python sederhana untuk melakukan checksum menggunakan MD5. Program tersebut menggunakan library bawaan bernama hashlib.

B. SHA-1

SHA-1 merupakan fungsi *hash cryptographic* yang menerima input lalu akan mengeluarkan output 160-bit (20 byte) nilai hash yang dikenal sebagai *message digest*, yang biasanya adalah angka hexadesimal dengan panjang 40 digit. SHA-1 dibuat oleh NSA, badan keamanan Amerika. Sejak tahun 2005, SHA sudah tidak dianggap aman oleh badan-badan keamanan [51]. NIST secara formal tidak memperbolehkan penggunaan SHA pada tahun 2013. Penggantian SHA-1 terbilang urgent pada penggunaannya dalam digital signature seperti SSL certificate. Semua web browser populer tidak menerima sertifikat SSL SHA-1 pada tahun 2017 [52, 53]. SHA-1 memproduksi message digest dan prinsipnya mirip dengan MD5, namun membuat nilai hash yang lebih besar (160 bit vs 128 bit). Version control system seperti Git, Mercurial, dan Monotone menggunakan SHA-1 bukan untuk sekuriti, namun untuk pengecekan revisi dan untuk menjamin datanya tidak berubah karena korup yang tidak disengaja.

Untuk hash function yang mana L adalah jumlah bit di message digest, menemukan message yang sesuai dengan message digest dapat selalu dilakukan dengan metode *brute force* yang kira-kira dilakukan 2^L iterasi. Ini dinamakan preimage attack dan mungkin bekerja mungkin juga tidak tergantung dari L dan kekuatan komputasi dari komputer. Akan tetapi, sebuah *collision*, yang terdiri dari menemukan dua pesan berbeda yang menghasilkan message digest yang sama memerlukan kurang lebih $1.2 \times 2^{L/2}$ iterasi apabila menggunakan birthday attack. Beberapa aplikasi yang menggunakan cryptographic hashes, seperti penyimpanan password, tidak terlalu terdampak oleh collision attack. Membuat password yang bekerja untuk suatu akun membutuhkan preimage attack, dan juga akses ke hash pada password originalnya. Pada kasus tanda tangan dokumen, attacker tidak dapat langsung memalsukan signature dari dokumen yang sudah ada. Attacker harus membuat sepasang dokumen, satu yang tidak merusak, dan satu untuk merusak, dan membuat pemegang *private key* untuk menandatangani dokumen yang tidak merusak.

```
import hashlib

file_name = 'filename.exe'
```

```
original_sha1 = ('d1e67b8819b009ec6942033'
                 'b6fc1928dd64b5df31bcde63'
                 '81b9d3f90488d25324049046'
                 '0c0a5a1a873da8236c12ef969')

with open(file_name) as file_to_check:
    data = file_to_check.read().encode()
    sha1_returned = hashlib.sha1(data).hexdigest()

if original_sha1 == sha1_returned:
    print('SHA-1 verified.')
else:
    print('SHA-1 verification failed!.')
```

Listing 4. Program python sederhana untuk cek sha1sum

Pada Listing 4 adalah contoh program python sederhana untuk melakukan checksum menggunakan algoritma SHA-1. Program tersebut menggunakan library bawaan bernama hashlib.

C. SHA-2

SHA-2 merupakan perubahan yang cukup signifikan dibandingkan pendahulunya, SHA-1. SHA-2 family terdiri atas enam buah hash function dengan digests (nilai hash) yang bernilai 2254, 256, 384 atau 512 bit: SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224, SHA-512/256. Hash function pada SHA-2 digunakan secara luas pada sejumlah aplikasi dan protokol sekuriti seperti TLS dan SSL, PGP, SSH, S/MIME dan IPsec. SHA-256 digunakan dalam proses autentikasi package software di OS Debian. SHA-256 dan SHA-512 direncanakan untuk digunakan pada DNSSEC [54]. Makin besar nilai hashnya maka artinya semakin aman dan sulit untuk menemukan collision, tetapi membutuhkan beban komputasi yang lebih tinggi.

D. SHA-3

SHA-3 merupakan member terbaru dari family Secure Hash Algorithm (SHA), yang dirilis oleh NIST pada tahun 2015. Meskipun masih satu family, SHA-3 secara internal berbeda dengan SHA-1 dan SHA-2 yang strukturnya seperti MD5. SHA-3 merupakan subset dari *Keccak* [55].

Keccak didasari dari pendekatan baru yang dinamakan sponge construction. Sponge construction didasari dari fungsi random atau fungsi permutasi, dan menerima (atau "menyerap", dalam istilah sponge) input data sebanyak apapun, kemudian mengoutput ("memeras") berapapun jumlah datanya, sambil bertindak sebagai *pseudorandom function*. Pada state absorbing, blok pesan di-XOR menjadi subset dari state, yang kemudian ditransformasikan secara keseluruhan menggunakan fungsi permutasi f. Pada state "squeeze", blok-blok output dibaca dari subset yang sama dari state, berselang-seling dengan fungsi transformasi state f. Ukuran dari bagian pada state yang tertulis dan dibaca dinamakan "rate" (dinotasikan sebagai r), dan ukuran dari bagian yang tidak tersentuh oleh input/output dinamakan capacity (dinotasikan sebagai c). Kapasitas ini menentukan tingkat dari sekuritinya. Level maksimum dari sekuritinya adalah separuh dari kapasitasnya. Akgoritma dari *Keccak* adalah sebagai berikut [56].

Fungsi theta terdiri dari tiga persamaan yang melibatkan operasi XOR sederhana dan bitwise cyclic shift.

$$C[X] = XOR(A[X,0], A[X,1], A[X,2], A[X,3], A[X,4]) \tag{5}$$

Circular shift ke kiri pertama diaktifkan di lima jalur output. Setelah tahap ini, jalur terakhir menjadi yang pertama dan jalur kedua terakhir akan menjadi jalur terakhir. Kemudian circular shift ke kanan dilakukan pada lajur tersebut sehingga jalur pertama menjadi jalur terakhir dan jalur kedua menjadi jalur pertama. Sekarang circular shift ke kiri diterapkan pada setiap jalur untuk mengubah posisi dari bit pada tiap lajur [56, 57].

$$D[X] = ROT(C[X-1], C[X+1,1]) \tag{6}$$

Input state matrix dan output jalur yang didapat dari (5)(6) di-XOR kan.

$$A[X,Y] = XOR(A[X,Y], D[X]), 0 \leq X; Y \leq 4 \tag{7}$$

Rho (ρ) dan Pi (Π) dapat diekspresikan untuk menghitung array B berukuran 5×5 dari state array A. Operasi dari rho dan Pi, melakukan rotasi sirkular sebanyak nilai tetap yang bergantun pada koordinat X dan Y.

$$r[X,Y], B[Y, 2X+3Y] = ROT(A[X,Y], r[X,Y]) \tag{8}$$

Jalur yang dirotasikan ditempatkan di array B. Ini disebut sebagai Pi step. Indeks yang diambil adalah modulo 5.

Jalurnya dioperasikan dengan fungsi chi (χ) seperti berikut.

$$A[X,Y] = B[X,Y]((\neg B[X+1,Y]) \wedge B[X+2,Y]) \tag{9}$$

Step Iota (10) adalah step paling sederhana dari algoritma Keccak. Step ini hanya melakukan operasi XOR pada konstanta bit RC dengan jalur pada lokasi $[0,0]$ dari matrix state baru A.

$$A[0,0] = XOR(A[0,0], RC) \tag{10}$$

```
import hashlib

file_name = 'filename.exe'

original_sha3 = ('3706a96a8fa96b3fc5ff30c'
                 'bca36ce666042e2d07762022'
                 'a78a2ec82439848fc3695e83'
                 '336ab71f47dddbc24b96454df2a43'
                 '7e343801a4e13faab89e8d0fda61')

with open(file_name) as file_to_check:
    data = file_to_check.read().encode()
    sha3_returned = hashlib.sha3_512(data).hexdigest()
```

```
if original_sha3 == sha3_returned:
    print('SHA-3 verified.')
else:
    print('SHA-3 verification failed!.')
```

Listing 5. Program python sederhana untuk cek sha3sum 512 bit

Program yang ada di listing 5 menunjukkan program python sederhana untuk melakukan pengecekan file integrity menggunakan algoritma SHA-3 512-bit.

Etiam euismod. Fusce facilisis lacinia dui. Suspendisse potenti. In mi erat, cursus id, nonummy sed, ullamcorper eget, sapien. Praesent pretium, magna in eleifend egestas, pede pede pretium lorem, quis consectetuer tortor sapien facilisis magna. Mauris quis magna varius nulla scelerisque imperdiet. Aliquam non quam. Aliquam porttitor quam a lacus. Praesent vel arcu ut tortor cursus volutpat. In vitae pede quis diam bibendum placerat. Fusce elementum convallis neque. Sed dolor orci, scelerisque ac, dapibus nec, ultricies ut, mi. Duis nec dui quis leo sagittis commodo.

V. Cyclic Redundancy Check

Sebuah *Cyclic Redundancy Check* adalah *error-detecting code* yang biasa digunakan pada jaringan digital dan storage untuk mendeteksi adanya perubahan yang tidak diinginkan pada data. Blok data yang memasuki sistem CRC akan diberikan *check value*, berdasarkan sisa dari pembagian polinomial dari kontennya. Saat pengambilan data, kalkulasi tersebut diulang lagi dan apabila check value tidak sesuai maka dapat dilakukan koreksi untuk menghindari data yang korup. CRC dapat digunakan untuk *error-correction* [58]. Varian CRC-1 dikenal juga sebagai parity bit (III).

Sejatinya, CRC merupakan tipe dari checksum, dan memiliki konsep yang mirip dengan checksum. Akan tetapi, terdapat perbedaan diantaranya sehingga saya memutuskan untuk memberikannya bab tersendiri. CRC merupakan checksum yang secara spesifik adalah *position dependent checksum algorithm*. Dari namanya tersebut, CRC dapat mendeteksi perpindahan posisi, yang membuatnya menjadi integrity check yang umum digunakan. CRC juga populer dikarenakan kesederhanaannya dibandingkan algoritma checksum yang lainnya seperti MD5 dan SHA family. CRC juga lebih mudah untuk dianalisis secara matematis dan baik untuk mendeteksi error yang umum terjadi dikarenakan oleh noise pada transmission channel. CRC sendiri tidak didesain dengan tujuan kriptografik, karena CRC dapat direverse sehingga untuk alasan keamanan lebih dianjurkan untuk menggunakan SHA-2. CRC biasa digunakan dalam hal untuk menyalin atau memindahkan file serta kompres dan dekompres file.

A. Integritas Data

Seperti yang sudah disebutkan sebelumnya, CRC didesain secara spesifik untuk keperluan error-checking, dimana CRC akan mendeteksi kesalahan dengan beban komputasi yang jauh lebih ringan dibandingkan dengan Cryptographic Hash Function. Maka dari itu, CRC tidak cocok untuk melindungi dari modifikasi data yang disengaja. Yang pertama, karena tidak ada autentikasi, *attacker* dapat memodifikasi pesan dan menghitung ulang CRCnya tanpa terdeteksi. Ketika disimpan bersama dengan data, baik CRC maupun Cryptographic Hash Function tidak melindungi dari perubahan data yang disengaja. Aplikasi yang memerlukan proteksi dari serangan tersebut harus menggunakan mekanisme autentikasi kriptografik, seperti *message authentication codes* (MAC) atau *digital signatures*. Yang kedua, tidak seperti MD5 maupun SHA, CRC dapat dengan mudah direverse, yang membuat CRC tidak cocok untuk digunakan sebagai *digital signatures* [59]. Yang ketiga, CRC memiliki hubungan yang mirip dengan fungsi linear [60].

$$CRC(x \oplus y) = CRC(x) \oplus CRC(y) \oplus c \tag{11}$$

Dimana c bergantung dari panjang x dan y. Persamaan 11 juga bisa dituliskan seperti berikut, dimana x, y, dan z memiliki panjang yang sama.

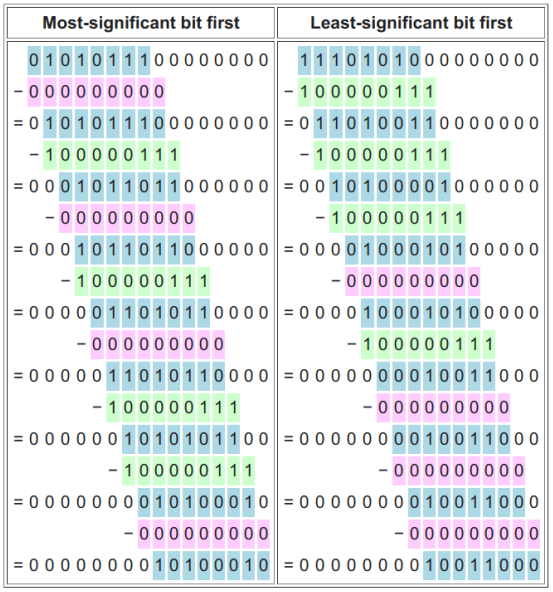
$$CRC(x \oplus y \oplus z) = CRC(x) \oplus CRC(y) \oplus CRC(z) \tag{12}$$

Maka, bahkan ketika CRC dienkripsi dengan *stream cipher* yang menggunakan XOR sebagai operasi kombinasinya, baik pesan maupun CRC dapat dimanipulasi tanpa sepengetahuan dari *encryption key*, ini merupakan salah satu *design flaws* dari protokol Wired Equivalent Privacy (WEP) [61].

B. Algoritma

Komputasi dari CRC diturunkan dari polynomial division, modulo dua. Ini menyerupai pembagian dari pesan string biner, dengan nol bit yang jumlahnya tetap diappend oleh string "generator polynomial", tetapi dengan menggunakan XOR, bukan pengurangan. Pembagian jenis ini sudah direalisasikan di harware dengan shift register yang sudah dimodifikasi [62]. Contoh dari implementasi polynomial division pada hardware, misalnya kita mencoba untuk menghitung CRC 8-bit dari pesan 8-bit yang terdiri dari karakter ASCII "W", yang binernya adalah 01010111_2, desimal 87_10 atau hexadecimal 57_16. Sebagai ilustrasi, kita menggunakan CRC-8-ATM (HEC) polinomial $x^6 + x^2 + x + 1$. Menuliskan bit pertama yang ditransmisikan (koefisien dari pangkat tertinggi x) di sebelah kiri, sesuai dengan string 9-bit "100000111". Nilai byte 57_16 dapat dikirimkan dalam dua urutan berbeda, bergantung dari konvensi urutan bit yang digunakan. Tiap urutan menghasilkan pesan polinomial $M(x)$ yang berbeda. Msbit-first, $x^6 + x^4 + x^2 + x + 1$ = 01010111, bila lsbit-first, $x^7 + x^6 + x^5 + 3 + 1$ = 11101010. Nilai ini dapat dikalikan dengan x^8 untuk menghasilkan dua pesan 16-bit polinomial $x^8M(x)$. Menghitung sisanya terdiri dari mengurangi kelipatan dari generator polinomial $G(x)$. Ini hanya seperti pembagian desimal, tapi lebih sederhana karena kelipatan yang mungkin hanyalah 0 dan 1, dan pengurangannya meminjam dari "tak terhingga", bukan mengurangi dari digit yang lebih tinggi. Karena kita tidak peduli dengan hasil bagi, maka tidak perlu dituliskan.

Perhatikan bahwa untuk setiap pengurangan, bit-bitnya dibagi menjadi 3 bagian, grup yang berisi nol, grup yang tidak



Gambar 3. Perhitungan polynomial division

diubah dari originalnya, dan grup yang berwarna biru, yang "menarik". Grup yang menarik ini panjangnya 8-bit, menyamai pangkat dari polinomial. Setiap langkah, kelipatan yang benar dikurangi untuk membuat grup nol menjadi satu bit lebih besar, dan grup yang tidak berubah menjadi satu bit lebih pendek, hingga akhirnya meninggalkan satu sisa.

```
def crc_remainder(input_bitstring,
                  polynomial_bitstring,
                  initial_filler):
    """Calculate the CRC remainder of a string of
    bits using a chosen polynomial.
    initial_filler should be '1' or '0'.
    """
    polynomial_bitstring = polynomial_bitstring.lstrip('0')
    len_input = len(input_bitstring)
    initial_padding = (len(polynomial_bitstring) - 1) \
         * initial_filler
    input_padded_array = list(input_bitstring \
        + initial_padding)
    while '1' in input_padded_array[:len_input]:
        cur_shift = input_padded_array.index('1')
        for i in range(len(polynomial_bitstring)):
            input_padded_array[cur_shift + i] \
            = str(\
                int(polynomial_bitstring\
                    [i] != input_padded_array\
            [cur_shift + i]))
    return ''.join(input_padded_array)[len_input:]

def crc_check(input_bitstring,
              polynomial_bitstring,
              check_value):
    """Calculate the CRC check of a string of bits using a
    chosen polynomial."""
    polynomial_bitstring = polynomial_bitstring.lstrip('0')
    len_input = len(input_bitstring)
    initial_padding = check_value
    input_padded_array = \
        list(input_bitstring + initial_padding)
    while '1' in input_padded_array[:len_input]:
        cur_shift = input_padded_array.index('1')
        for i in range(len(polynomial_bitstring)):
            input_padded_array[cur_shift + i] \
            = str(int(polynomial_bitstring\
                [i] != input_padded_array\
            [cur_shift + i]))
    return ('1' not in ''.join(input_padded_array)[len_input:])
```

Listing 6. Program CRC.

Program dari listing 6 berisi fungsi yang akan mereturn nilai sisa awal dari CRC untuk input dan polinomial yang ditentukan, entah dengan 1 atau 0 sebagai padding awalnya.

```
>>> crc_remainder('11010011101100', '1011', '0')
'100'
>>> crc_check('11010011101100', '1011', '100')
True
```

Listing 7. Output dari listing 6.

```
#include <inttypes.h> // uint32_t, uint8_t

uint32_t CRC32(const uint8_t data[], size_t data_length) {
        uint32_t crc32 = 0xFFFFFFFFu;

        for (size_t i = 0; i < data_length; i++) {
                const uint32_t lookupIndex =
                (crc32 ^ data[i]) & 0xff;
                crc32 =
                (crc32 >> 8) ^ CRCTable[lookupIndex];
// CRCTable is an array of 256 32-bit constants
        }

// Finalize the CRC-32 value
// by inverting all the bits
        crc32 ^= 0xFFFFFFFFu;
        return crc32;
}
```

Listing 8. Program CRC dalam bahasa C.

Listing 8 merupakan algoritma CRC-32 [63] dalam bahasa C. Variable `CRCTable` adalah memoization dari kalkulasi yang harus diulang untuk setiap byte dari pesan.

C. Kompresi Data

CRC digunakan untuk melakukan kalkulasi dari semua data yang ada di dalam file yang terkompresi. Nilai CRC akan dikalkulasi setiap kali ada file baru yang ditambahkan kedalam archive. Ketika archive atau file terkompresi di dekompresi, maka program akan mengkalkulasi nilai CRC kembali dan membandingkannya dengan yang ada di archive. Apabila

terdapat perbedaan pada CRC value, maka biasanya akan ditampilkan pesan CRC error, yang mengindikasikan bahwa file yang terekstrak tidak sama dengan file yang awalnya dikompresi. Hal ini biasanya terjadi ketika file yang dikompresi didalam archive rusak. Hal ini juga dapat terjadi walaupun hanya satu file didalam archive yang korup.

Nilai dari CRC itu sendiri tidak mengatakan bahwa file anda korup atau tidak. Maka, ketika kita mengecek metadata dari archive atau mengkompresi file dan menemukan nilai CRC-nya, bukan berarti archive kita telah rusak. Hal tersebut hanya menunjukkan nilai awal dari CRC file yang terkompresi. Kebanyakan software untuk melakukan kompresi seperti 7zip dan WinRAR sudah memiliki mekanisme ini [64].

VI. Kesimpulan

Dari semua bagian yang sudah diberikan pembahasannya pada makalah ini, dapat diberi kesimpulan sebagai berikut. Parity Bit adalah salah satu cara untuk melakukan uji integritas file yang paling sederhana, tetapi pada penggunaannya terdapat kelemahan-kelemahan seperti jumlah bit yang sama-sama genap maupun ganjil tetapi berbeda dari yang aslinya akan dianggap tidak benar. Parity bit sendiri memiliki kelebihan seperti tidak memerlukan beban komputasi yang besar. Checksum sendiri merupakan istilah yang lebih luas dari algoritma-algoritma yang digunakan untuk melakukan pengecekan integritas. Terdapat banyak algoritma untuk melakukan checksum, salah satunya yang populer adalah MD5. MD5 merupakan algoritma hash function yang umum digunakan, meski sekarang hanya digunakan untuk sekedar melakukan cek integritas pada file yang tidak dimodifikasi secara sengaja. Hal ini terjadi karena komputer sekarang makin cepat sehingga metode brute force untuk menemukan collision dapat dengan mudah dilakukan. MD5 adalah varian dari message digest algorithm. SHA-1 merupakan generasi kedua dari keluarga SHA (Secure Hash Algorithm). SHA-1 juga sudah ditinggalkan karena alasan keamanan yang sama dengan MD5. SHA-2 merupakan hash function algorithm yang paling umum digunakan sekarang. NIST masih merekomendasikan SHA-2, namun disarankan menggunakan hash value yang besar agar semakin aman. SHA-2 juga dapat digunakan untuk melakukan pengecekan integritas file, namun apabila tidak memedulikan aspek sekuriti tidak disarankan karena beban komputasi yang lebih tinggi. SHA-3 merupakan terobosan terbaru dari keluarga SHA, karena menggunakan pendekatan yang berbeda dari varian sebelumnya, Keccak Algorithm. SHA-3 diyakini lebih aman dalam masalah anti collision, namun belum secara luas digunakan. CRC atau Cyclic Redundancy Check merupakan salah satu jenis checksum namun tidak diperuntukkan untuk keamanan. CRC bagus digunakan untuk melakukan verifikasi saat menyalin file ataupun pengarsipan file. Algoritma yang digunakan CRC juga terbilang tidak serumit MD5 dan SHA, sehingga pada *Integrated Ciruit* biasanya sudah diaplikasikan dan siap digunakan.

Pustaka

[1] J. Hu and A. Lanzon, "An innovative tri-rotor drone and associated distributed aerial drone swarm control," *Robotics and Autonomous Systems*, vol. 103, pp. 162–174, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0921889017308163

[2] J. Cary, Leslie; Coyne, "Uas yearbook - uas: The global perspective," *ICAO Unmanned Aircraft Systems (UAS), Circular 328*, 2011-2012.

[3] "Unmanned aerial vehicles – the force multiplier of the 1990s," 2009. [Online]. Available: https://web.archive.org/web/20090724015052/http://www.airpower.maxwell.af.mil/airchronicles/apj/apj91/spr91/4spr91.htm

[4] J. Hu, P. Bhowmick, I. Jang, F. Arvin, and A. Lanzon, "A decentralized cluster formation containment framework for multirobot systems," *IEEE Transactions on Robotics*, vol. 37, no. 6, pp. 1936–1955, 2021.

[5] C. Koparan, A. B. Koc, C. V. Privette, and C. B. Sawyer, "Adaptive water sampling device for aerial robots," *Drones*, vol. 4, no. 1, 2020. [Online]. Available: https://www.mdpi.com/2504-446X/4/1/5

[6] C. Koparan, A. B. Koc, C. V. Privette, C. B. Sawyer, and J. L. Sharp, "Evaluation of a uav-assisted autonomous water sampling," *Water*, vol. 10, no. 5, 2018. [Online]. Available: https://www.mdpi.com/2073-4441/10/5/655

[7] C. Koparan, A. B. Koc, C. V. Privette, and C. B. Sawyer, "In situ water quality measurements using an unmanned aerial vehicle (uav) system," *Water*, vol. 10, no. 3, 2018. [Online]. Available: https://www.mdpi.com/2073-4441/10/3/264

[8] c. koparan, a. b. koc, c. v. privette, and c. b. sawyer, "autonomous in situ measurements of noncontaminant water quality indicators and sample collection with a uav," *water*, vol. 11, no. 3, 2019. [Online]. Available: https://www.mdpi.com/2073-4441/11/3/604

[9] "Drones smuggling porn, drugs to inmates around the world," 2017. [Online]. Available: http://www.foxnews.com/us/2017/04/17/drones-smuggling-porn-drugs-to-inmates-around-world.html

[10] M. Ayamga, S. Akaba, and A. A. Nyaaba, "Multifaceted applicability of drones: A review," *Technological Forecasting and Social Change*, vol. 167, p. 120677, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0040162521001098

[11] L. Ramadass, S. Arunachalam, and Z. Sagayasree, "Applying deep learning algorithm to maintain social distance in public place through drone technology," *Int. J. Pervasive Comput. Commun.*, vol. 16, pp. 223–234, 2020.

[12] B. McCall, "Sub-saharan africa leads the way in medical drones," *The Lancet*, vol. 393, pp. 17–18, 01 2019.

[13] R. Kellermann, T. Biehle, and L. Fischer, "Drones for parcel and passenger transportation: A literature review," *Transportation Research Interdisciplinary Perspectives*, vol. 4, p. 100088, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2590198219300879

[14] B. Bolo, D. Mpoeleng, and I. Zlotnikova, "Development of methods acquiring real time very high resolution agricultural spatial information using unmanned aerial vehicle," *Agris on-line Papers in Economics and Informatics*, vol. 11, pp. 21–29, 06 2019.

[15] R. A. Clothier, D. A. Greer, D. G. Greer, and A. M. Mehta, "Risk perception and the public acceptance of drones," *Risk Analysis*, vol. 35, no. 6, pp. 1167–1183, 2015. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/risa.12330

[16] G. Sylvester, *E-agriculture in action: Drones for agriculture*. Food and Agriculture Organization ofn the United Nations and International ..., 2018.

[17] M. Gharibi, R. Boutaba, and S. L. Waslander, "Internet of drones," *IEEE Access*, vol. 4, pp. 1148–1162, 2016.

[18] G. Choudhary, V. Sharma, T. Gupta, J. Kim, and I. You, "Internet of drones (iod): threats, vulnerability, and security perspectives," *arXiv preprint arXiv:1808.00203*, 2018.

[19] S. Times, "Food delivery via drones in cyberjaya by end of the month," *Accessed: Apr*, vol. 4, p. 2020, 2020.

[20] A. Yazdinejad, R. M. Parizi, A. Dehghantanha, H. Karimipour, G. Srivastava, and M. Aledhari, "Enabling drones in the internet of things with decentralized blockchain-based security," *IEEE Internet of Things Journal*, vol. 8, no. 8, pp. 6406–6415, 2020.

[21] A. D. Boursianis, M. S. Papadopoulou, P. Diamantoulakis, A. Liopa-Tsakalidi, P. Barouchas, G. Salahas, G. Karagiannidis, S. Wan, and S. K. Goudos, "Internet of things (iot) and agricultural unmanned aerial vehicles (uavs) in smart farming: a comprehensive review," *Internet of Things*, p. 100187, 2020.

[22] S. Magistretti and C. Dell'Era, "Unveiling opportunities afforded by emerging technologies: Evidences from the drone industry," *Technology Analysis & Strategic Management*, vol. 31, no. 5, pp. 606–623, 2019.

[23] D. Paddeu, T. Calvert, B. Clark, and G. Parkhurst, "New technology and automation in freight transport and handling systems," 2019.

[24] E. Vattapparamban, I. Güvenç, A. I. Yurekli, K. Akkaya, and S. Uluağaç, "Drones for smart cities: Issues in cybersecurity, privacy, and public safety," in *2016 international wireless communications and mobile computing conference (IWCMC)*. IEEE, 2016, pp. 216–221.

[25] M. Półka, S. Ptak, and Ł. Kuziora, "The use of uav's for search and rescue operations," *Procedia engineering*, vol. 192, pp. 748–752, 2017.

[26] V. Kharchenko and V. Torianyk, "Cybersecurity of the internet of drones: Vulnerabilities analysis and imeca based assessment," in *2018 IEEE 9th International Conference on Dependable Systems, Services and Technologies (DESSERT)*. IEEE, 2018, pp. 364–369.

[27] F. Thiobane, "Cybersecurity and drones," Ph.D. dissertation, Utica College, 2015.

[28] M. Rodrigues, J. Amaro, F. S. Osório, and B. K. RLJC, "Authentication methods for uav communication," in *2019 IEEE Symposium on Computers and Communications (ISCC)*. IEEE, 2019, pp. 1210–1215.

[29] M. F. B. A. Rahman, "Smart cctvs for secure cities: Potentials and challenges," 2017.

[30] M. Mohan, "Cybersecurity in drones," Ph.D. dissertation, Utica College, 2016.

[31] M. Yahuza, M. Y. I. B. Idris, A. W. B. A. Wahab, A. T. Ho, S. Khan, S. N. B. Musa, and A. Z. B. Taha, "Systematic review on security and privacy requirements in edge computing: State of the art and future research opportunities," *IEEE Access*, vol. 8, pp. 76 541–76 567, 2020.

[32] R. Altawy and A. M. Youssef, "Security, privacy, and safety aspects of civilian drones: A survey," *ACM Transactions on Cyber-Physical Systems*, vol. 1, no. 2, pp. 1–25, 2016.

[33] D. He, S. Chan, and M. Guizani, "Drone-assisted public safety networks: The security aspect," *IEEE Communications Magazine*, vol. 55, no. 8, pp. 218–223, 2017.

[34] M. Yampolskiy, P. Horvath, X. D. Koutsoukos, Y. Xue, and J. Sztipanovits, "Taxonomy for description of cross-domain attacks on cps," in *Proceedings of the 2nd ACM international conference on High confidence networked systems*, 2013, pp. 135–142.

[35] H. Sedjelmaci and S. M. Senouci, "Cyber security methods for aerial vehicle networks: taxonomy, challenges and solution," *The Journal of Supercomputing*, vol. 74, no. 10, pp. 4928–4944, 2018.

[36] T. Humphreys, "Statement on the vulnerability of civil unmanned aerial vehicles and other systems to civil gps spoofing," *University of Texas at Austin (July 18, 2012)*, pp. 1–16, 2012.

[37] D. P. Shepard, J. A. Bhatti, T. E. Humphreys, and A. A. Fansler, "Evaluation of smart grid and civilian uav vulnerability to gps spoofing attacks," in *Proceedings of the 25th International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS 2012)*, 2012, pp. 3591–3605.

[38] I. Güvenç, O. Ozdemir, Y. Yapici, H. Mehrpouyan, and D. Matolak, "Detection, localization, and tracking of unauthorized uas and jammers," in *2017 IEEE/AIAA 36th Digital Avionics Systems Conference (DASC)*. IEEE, 2017, pp. 1–10.

[39] R. L. Sturdivant and E. K. Chong, "Systems engineering baseline concept of a multispectral drone detection solution for airports," *IEEE Access*, vol. 5, pp. 7123–7138, 2017.

[40] X. Shi, C. Yang, W. Xie, C. Liang, Z. Shi, and J. Chen, "Anti-drone system with multiple surveillance technologies: Architecture, implementation, and challenges," *IEEE Communications Magazine*, vol. 56, no. 4, pp. 68–74, 2018.

[41] B. Nassi, A. Shabtai, R. Masuoka, and Y. Elovici, "Sok - security and privacy in the age of drones: Threats, challenges, solution mechanisms, and scientific gaps," 03 2019.

[42] J. Yao and N. Ansari, "Qos-aware power control in internet of drones for data collection service," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 7, pp. 6649–6656, 2019.

[43] C. Lin, D. He, N. Kumar, K.-K. R. Choo, A. Vinel, and X. Huang, "Security and privacy for the internet of drones: Challenges and solutions," *IEEE Communications Magazine*, vol. 56, no. 1, pp. 64–69, 2018.

[44] S. Aggarwal, M. Shojafar, N. Kumar, and M. Conti, "A new secure data dissemination model in internet of drones," in *ICC 2019-2019 IEEE International Conference on Communications (ICC)*. IEEE, 2019, pp. 1–6.

[45] B. Qureshi, A. Koubâa, M.-F. Sriti, Y. Javed, and M. Alajlan, "Dronemap-a cloud-based architecture for the internet-of-drones," in *International Conference on Embedded Wireless Systems and Networks*, 2016.

[46] W. H. RodgerE, Ziemer; Tranter, *Principles of communication : systems, modulation, and noise*. New Jersey: Hoboken, 2015.

[47] B. Preneel, "Cryptographic hash functions," *Transactions on Emerging Telecommunications Technologies*, 1994.

[48] "Nist policy on hash functions," 2015. [Online]. Available: https://csrc.nist.gov/projects/hash-functions/nist-policy-on-hash-functions

[49] M. Kleppmann, *Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems*, 1st ed. O'Reilly Media, 2017.

[50] "A quarter of major cmss use outdated md5 as the default password hashing scheme," 2019. [Online]. Available: https://www.zdnet.com/article/a-quarter-of-major-cmss-use-outdated-md5-as-the-default-password-hashing-scheme/

[51] "Schneier on security: Cryptanalysis of sha-1," 2005. [Online]. Available: https://www.schneier.com/blog/archives/2005/02/cryptanalysis_o.html

[52] "The end of sha-1 on the public web," 2019. [Online]. Available: https://blog.mozilla.org/security/2017/02/23/the-end-of-sha-1-on-the-public-web/

[53] "Google will drop sha-1 encryption from chrome by january 1, 2017," 2015. [Online]. Available: https://venturebeat.com/2015/12/18/google-will-drop-sha-1-encryption-from-chrome-by-january-1-2017/

[54] "Use of sha-2 algorithms with rsa in dnskey and rrsig resource records for dnssec," 2009. [Online]. Available: https://www.rfc-editor.org/rfc/rfc5702.txt

[55] "Nist selects winner of secure hash algorithm (sha-3) competition," 2012. [Online]. Available: https://www.nist.gov/itl/csd/sha-100212.cfm

[56] A. Arshad, D.-e.-S. Kundi, and A. Aziz, "Compact implementation of sha3-512 on fpga," in *2014 Conference on Information Assurance and Cyber Security (CIACS)*, 2014.

[57] N. Chandran and E. Manuel, "Performance analysis of modified sha-3," *Procedia Technology*, vol. 24, pp. 904–910, 12 2016.

[58] "An algorithm for error correcting cyclic redundance checks7," 2003. [Online]. Available: https://www.drdobbs.com/an-algorithm-for-error-correcting-cyclic/184401662

[59] M. Stigge, H. Plotz, W. Muller, and J.-P. Redlich, "Reversing crc(theory and practice)," in *HU Berlin Public Report SAR-PR-2006-05*, 2006. [Online]. Available: http://sar.informatik.hu-berlin.de/research/publications/SAR-PR-2006-05/SAR-PR-2006-05_.pdf

[60] "algorithm design - why is crc said to be linear?" 2003. [Online]. Available: https://crypto.stackexchange.com/a/34013

[61] N. Cam-Winget, R. Housley, D. Wagner, and J. Walker, "Security flaws in 802.11 data link protocols," *Commun. ACM*, vol. 46, no. 5, p. 35–39, May 2003. [Online]. Available: https://doi.org/10.1145/769800.769823

[62] E. Dubrova and S. S. Mansouri, "A bdd-based approach to constructing lfsrs for parallel crc encoding," in *2012 IEEE 42nd International Symposium on Multiple-Valued Logic*, 2012, pp. 128–133.

[63] "[ms-abs]: 32-bit crc algorithm | microsoft docs," 2019. [Online]. Available: https://msdn.microsoft.com/en-us/library/dd905031.aspx

[64] "How to recover corrupted 7z archive." [Online]. Available: https://www.7-zip.org/recover.html