Penerapan Enkripsi Homomorfik dan K-Anonymity untuk Deteksi Password Bocor

Wisyendra Lunarmalam - 18222095
Program Studi Sistem dan Teknologi Informasi
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: lunarwisyendra@gmail.com , 18222095@std.stei.itb.ac.id

Kebocoran data yang mengandung informasi kredensial seperti kata sandi merupakan ancaman serius bagi keamanan dan privasi digital. Have I Been Pwned (HIBP) adalah salah satu layanan populer yang memungkinkan pengguna memeriksa apakah kata sandi mereka telah bocor. Untuk menjaga privasi, HIBP menggunakan skema k-anonymity, di mana hanya sebagian dari hash kata sandi yang dikirim ke server, sehingga server tidak dapat langsung mengidentifikasi kata sandi asli.

Meskipun skema k-anonymity ini mengurangi risiko kebocoran secara langsung, pendekatan tersebut masih memiliki kelemahan dalam hal perlindungan kriptografis selama proses pencocokan data. Oleh karena itu, makalah ini mengusulkan metode alternatif yang mengombinasikan enkripsi homomorfik dengan k-anonymity untuk meningkatkan keamanan dalam proses verifikasi kata sandi yang telah bocor. Enkripsi homomorfik memungkinkan pencocokan dilakukan dalam bentuk terenkripsi, sehingga isi kata sandi tidak pernah terungkap di sisi server.

Tujuan utama dari makalah ini adalah untuk menelaah dan mengevaluasi penerapan enkripsi homomorfik dalam sistem pemeriksaan kebocoran kata sandi seperti HIBP, guna menghasilkan solusi yang lebih kuat dalam menjaga privasi pengguna. Pendekatan ini diharapkan mampu menutupi kelemahan dari penggunaan k-anonymity secara tunggal, tanpa mengorbankan efisiensi maupun akurasi sistem.

Keywords—Enkripsi homomorfik, Have I Been Pwned, K-anonymity

I. PENDAHULUAN

Dalam era digital ini, insiden kebocoran data menjadi salah satu tantangan utama dalam menjaga keamanan dan privasi pengguna. Salah satu data yang paling rentan dan bernilai tinggi adalah kredensial pengguna terutama email dan password. Data tersebut sering diperjualbelikan ketika terjadi kebocoran untuk melakukan serangan lanjutan seperti phising atau pencurian identitas.

Layanan *Have I Been Pwned* (HIBP) menyediakan cara bagi pengguna untuk memeriksa apakah akun atau password mereka telah bocor atau belum. Untuk mengurangi risiko pelanggaran privasi, HIBP telah mengimplementasikan *k-anonymity* di mana hanya sebagian dari hash password yang dikirimkan ke server. Meskipun demikian, skema *k-anonymity* tidak sepenuhnya menghilangkan risiko kebocoran informasi parsial. Penyerang yang memiliki pengetahuan atau akses tambahan tetap dapat melakukan analisis atau serangan brute force. Oleh karena itu, diperlukan pendekatan yang lebih kuat untuk melindungi privasi pengguna, terutama selama proses pencocokan terhadap database.

Salah satu solusi potensial adalah penggunaan enkripsi homomorfik, yaitu skema kriptografi yang memungkinkan operasi dilakukan langsung pada data terenkripsi, tanpa perlu mendekripsinya terlebih dahulu. Dengan pendekatan ini, pengguna dapat mengenkripsi hash password mereka, dan server dapat melakukan pencocokan terhadap database bocor tanpa pernah melihat nilai sebenarnya. Kombinasi antara enkripsi homomorfik dan k-anonymity diharapkan dapat meningkatkan keamanan sistem pengecekan password secara signifikan.

Makalah ini berfokus pada eksplorasi dan evaluasi penerapan enkripsi homomorfik dalam konteks sistem seperti HIBP, sebagai upaya untuk memberikan solusi verifikasi password bocor yang lebih aman dan *privacy-preserving*.

II. PEMBAHASAN

A. K-Anonymity

K-anonymity adalah sebuah teknik pelindung privasi dalam publikasi atau pertukaran data, yang bertujuan untuk menyembunyikan identitas individu dengan membuat data tidak dapat dibedakan dari setidaknya k - 1 individu lainnya. Konsep ini terutama digunakan dalam konteks data publishing untuk melindungi *quasi-identifiers* atribut seperti umur, kode pos, atau jenis kelamin yang, jika digabungkan, dapat mengidentifikasi individu.

Dalam layanan seperti *Have I Been Pwned* (HIBP), *k-anonymity* diterapkan melalui penggunaan teknik hash parsial. Ketika pengguna ingin memeriksa apakah password mereka pernah bocor, hanya 5 karakter awal dari hash SHA-1 password yang dikirimkan ke server. Server kemudian mengembalikan daftar hash yang memiliki prefix yang sama, dan pencocokan akhir dilakukan di sisi klien. Hal ini membuat server tidak pernah mengetahui hash lengkap atau password asli pengguna. Metode ini memastikan bahwa permintaan dari satu pengguna tidak dapat secara langsung dikaitkan dengan

satu hash unik, karena setiap prefix mengembalikan setidaknya *k* (biasanya ratusan) kemungkinan.

Kelebihan dari penggunaan *k-anonymity* adalah privasi yang terjaga karena server tidak pernah menerima password atau hash utuh, efisien karena tidak memerlukan enkripsi atau proses kriptografi yang berat, dan cocok untuk digunakan pada sistem dengan banyak permintaan. Walaupun k-anonymity mengurangi risiko kebocoran identitas langsung, terdapat beberapa kelemahan seperti re-identifikasi yang masih mungkin jika data tambahan diketahui oleh penyerang.

Untuk menutupi kelemahan k-anonymity, pendekatan kriptografi seperti enkripsi homomorfik dapat digunakan untuk memungkinkan pencocokan secara aman terhadap data yang sepenuhnya terenkripsi, tanpa harus mengungkap hash secara parsial. Dengan cara ini, pencocokan hash dapat dilakukan oleh server terhadap database bocor secara langsung pada data terenkripsi, tanpa membuka konten hash tersebut.

B. *Enkripsi Homomorfik*

Enkripsi homomorfik (homomorphic encryption) adalah sebuah skema kriptografi yang memungkinkan perhitungan dilakukan langsung pada data yang telah dienkripsi, tanpa perlu mendekripsinya terlebih dahulu. Hasil dari perhitungan pada ciphertext, setelah didekripsi, akan sama dengan hasil yang diperoleh jika operasi dilakukan pada plaintext. Jika Enc(x) dan Enc(y) adalah enkripsi dari dua nilai x dan y, maka sistem homomorfik memungkinkan $Enc(x) \circ Enc(y) = Enc(x \oplus y)$ di mana $\circ$ adalah operasi pada ciphertext, dan $\oplus$ adalah operasi ekuivalen pada plaintext.

Dalam skema seperti *Have I Been Pwned*, pengguna mengirim sebagian hash (prefix) ke server untuk mencocokkan apakah password mereka bocor. Meskipun ini menjaga sebagian privasi (k-anonymity), data hash masih terekspos secara parsial. Dengan enkripsi homomorfik, pengguna dapat:

- Mengenkripsi password dengan kunci publik.
- Mengirim ciphertext ke server.
- Server melakukan pencocokan terhadap database hash bocor yang juga telah dienkripsi, tanpa pernah mendekripsi hash pengguna.
- Server mengembalikan hasil yang tetap dalam bentuk terenkripsi.
- Pengguna mendekripsi hasil untuk mengetahui apakah password-nya bocor.

Maka, tidak ada bagian dari password atau hash yang terlihat oleh server dan pencocokan dapat dilakukan secara privat dan aman.

Keuntungan dari aplikasi enkripsi homomorfik adalah data yang tetap terenkripsi di seluruh proses, tidak ada plaintext hash atau password, dan cocok untuk sistem *zero-trust*. Sementara kekurangan dari enkripsi homomorfik adalah performa dari enkripsi homomorfik sangat lambat dibanding metode biasa dan ciphertext bisa jauh lebih besar dari plaintext.

III. IMPLEMENTASI

A. *Arsitektur Sistem*

Sistem dirancang agar pengguna dapat memeriksa apakah password mereka termasuk dalam database password bocor tanpa membocorkan password itu sendiri, bahkan kepada server. Arsitektur akan terdiri dari 2 komponen, yaitu klien dan server. Klien akan bertanggung jawab untuk mengenkripsi hash password pengguna menggunakan skema enkripsi homomorfik. Server akan menyimpan database hash password bocor yang juga telah dienkripsi. Server melakukan pencocokan langsung terhadap ciphertext tanpa mendekripsinya. Proses yang terjadi:

1. Pengguna menginput password.
2. Password di-hash, lalu hasil hash dienkripsi menggunakan kunci publik homomorfik.
3. Ciphertext dikirim ke server.
4. Server melakukan pencocokan dengan ciphertext hash bocor yang telah disiapkan.
5. Server mengembalikan hasil yang masih terenkripsi.
6. Klien mendekripsi hasil dan menampilkan apakah password tersebut pernah bocor.

B. *Skema Enkripsi*

Untuk implementasi awal, digunakan Microsoft SEAL. *Library open-source* untuk enkripsi homomorfik berbasis BFV (Brakerski/Fan-Vercauteren scheme) dengan Fully Homomorphic Encryption (FHE).

C. *Implementasi Menggunakan Python*

Untuk menunjukkan bagaimana enkripsi homomorfik dapat digunakan dalam konteks layanan seperti Have I Been Pwned (HIBP), dilakukan implementasi sistem sederhana yang memungkinkan pengguna memverifikasi apakah password-nya termasuk dalam daftar password yang telah bocor, tanpa mengungkapkan informasi sensitif baik kepada server maupun pihak ketiga. Implementasi ini menggunakan skema BFV (Brakerski/Fan-Vercauteren) yang disediakan oleh *library* TenSEAL.

D. *Langkah-langkah implementasi*

1. Inisiasi konteks enkripsi
   Generate kunci menggunakan skema BFV dengan poly modulus degree sebesar 4096 dan plain modulus sebesar 1032193. 4096 dipilih sebagai angka poly modulus degree karena itu menentukan ukuran vektor ciphertext, 4096 adalah angka paling kecil yang dianggap aman sehingga cocok untuk demonstrasi. Sementara 1032193 dipilih sebagai plain modulus karena itu mengatur rentang angka plaintext yang bisa dienkripsi dan 1032193 adalah bilangan prima yang cukup besar untuk menampung nilai hasil enkripsi. Lalu generate kunci galois dan relin untuk operasi aritmatika homomorfik dan simpan kunci rahasia untuk dekripsi pada sisi klien.

```
import tenseal as ts
import hashlib

context = ts.context(
        ts.SCHEME_TYPE.BFV,
        poly_modulus_degree=4096,
        plain_modulus=1032193
)
context.generate_galois_keys()
context.generate_relin_keys()

secret_key = context.secret_key()
context.make_context_public()
```

2. Hash password

Hash password disini menggunakan SHA-1 dan hanya 8 karakter awal dari hasil hash yang diambil untuk mensimulasikan cara kerja HIBP.

```
def hash_password(password):
        sha1 = hashlib.sha1()
        sha1.update(password.encode())
        return int(sha1.hexdigest()[:8], 16)
```

3. Enkripsi password bocor

Simulasi daftar password bocor dilakukan dengan 3 data dummy, yaitu “123456”, “password”, dan “qwerty”. Hash dari setiap password akan dienkripsi dengan konteks yang telah dibuat.

```
leaked_passwords = ["123456", "password", "qwerty"]
leaked_hashes = [hash_password(p) for p in leaked_passwords]
leaked_encrypted = [ts.bfv_vector(context, [h]) for h in leaked_hashes]
```

4. Enkripsi password pengguna

Password yang ingin diverifiskasi pengguna juga di-hash dan dienkripsi.

```
user_password = "password"
user_hash = hash_password(user_password)
user_encrypted = ts.bfv_vector(context, [user_hash])
```

5. Pencocokan homomorfik

Setiap hash bocor yang telah dienkripsi dibandingkan dengan hash password pengguna yang telah dienkripsi. Pencocokan dilakukan melalui operasi pengurangan homomorfik antara dua ciphertext. Jika hasil pengurangan setelah dekripsi (dilakukan di sisi klien) adalah nol, berarti password tersebut cocok dan telah ditemukan dalam daftar bocor.

```
match_found = False
for enc_hash in leaked_encrypted:
        diff = user_encrypted - enc_hash
        diff_plain = diff.decrypt(secret_key=secret_key)
        if diff_plain[0] == 0:
                match_found = True
                break
```

6. Hasil eksekusi

Jika password ditemukan, maka akan ditampilkan “Password ditemukan”, dan jika tidak akan ditampilkan “Password tidak ditemukan”

```
24    user_password = "password"

PROBLEMS    OUTPUT    DEBUG CONSOLE    TE

❯ python3 main.py
Password ditemukan
```

```
24    user_password = "castorice"

PROBLEMS    OUTPUT    DEBUG CONSOLE    TE

❯ python3 main.py
Password tidak ditemukan
```

IV. Hasil dan Analisis

A. *Hasil Eksperimen*

Implementasi yang dilakukan berhasil menunjukkan bahwa enkripsi homomorfik dapat digunakan untuk memeriksa apakah sebuah password termasuk dalam daftar password yang bocor, tanpa mengorbankan privasi pengguna. Pada saat pengguna memberikan input berupa password, sistem:

1. Mengubah password menjadi hash SHA-1, kemudian mengambil 8 digit awalnya.
2. Mengenkripsi hash tersebut menggunakan skema BFV.
3. Melakukan operasi pengurangan homomorfik terhadap seluruh password bocor yang telah dienkripsi sebelumnya.
4. Mengembalikan hasil operasi untuk didekripsi di sisi klien, guna mengetahui apakah ada kecocokan.

Sebagai contoh, ketika password "password" digunakan sebagai input, sistem mendeteksi kecocokan karena hash-nya ditemukan dalam database bocor. Sebaliknya, ketika password yang tidak ada dalam database digunakan, sistem tidak menunjukkan adanya kecocokan.

B. *Analisis Keamanan*

Pendekatan ini memiliki beberapa keunggulan dari sisi keamanan dibandingkan pendekatan k-anonymity yang digunakan oleh *Have I Been Pwned*:

Table 1

Aspek	K-Anonymity	Enkripsi Homomorfik
Data yang dikirim	5 digit pertama hash	Hash yang dienkripsi
Kemungkinan terjadinya deanonymisasi	Mungkin dengan brute force	Tidak mungkin tanpa adanya kunci kriptografi
Server mengetahui hash	5 digit pertama	Tidak sama sekali

Dengan demikian, sistem berbasis enkripsi homomorfik memberikan jaminan privasi yang lebih kuat, karena tidak ada bagian dari hash atau password yang dikirim dalam bentuk plaintext ke server.

C. Analisis Kinerja

Pendekatan ini memiliki beberapa keterbatasan terutama dalam sisi performa karena ukuran ciphertext menjadi lebih besar daripada plaintext dan operasi aritmatika homomorfik pada enkripsi dan pengurangan akan membutuhkan waktu yang lebih lama jika dibandingkan dengan pencocokan biasa. Selain itu akan dibutuhkan juga optimasi pada server jika ingin diaplikasikan pada *Have I Been Pwned*.

Walaupun unggul dalam aspek keamanan, pendekatan ini memiliki beberapa keterbatasan dan tantangan dari sisi performa, antara lain:

a. Kompleksitas Komputasi
Operasi aritmatika homomorfik jauh lebih berat dibandingkan operasi logika biasa. Pencocokan satu password membutuhkan beberapa detik, tergantung jumlah entri dalam database bocor dan kompleksitas enkripsi.

b. Ukuran Ciphertext
Ciphertext yang dihasilkan oleh skema homomorfik bisa berukuran 10–100 kali lebih besar dari plaintext. Hal ini berdampak pada penyimpanan (storage), transfer data (bandwidth), dan memori (RAM) saat pemrosesan.

c. Konsumsi Sumber Daya
Diperlukan perangkat klien dengan kemampuan komputasi yang memadai untuk mengenkripsi dan mendekripsi dengan cepat. Untuk aplikasi mobile atau edge devices, pendekatan ini masih memerlukan optimasi performa.

d. Solusi Potensial
Beberapa solusi untuk mengatasi keterbatasan performa antara lain:

Parallelization: Melakukan pencocokan secara paralel pada beberapa core CPU atau GPU.

Batched Encoding: Memanfaatkan fitur batching dari skema BFV untuk mencocokkan banyak hash dalam satu operasi.

Caching: Menyimpan ciphertext hash bocor yang telah dienkripsi secara offline agar tidak perlu dienkripsi ulang setiap waktu.

Hybrid Approach: Menggabungkan k-anonymity dan enkripsi homomorfik hanya untuk pengguna yang memilih mode privacy enhanced.

D. Analisis Skalabilitas

Untuk penerapan nyata pada layanan seperti *Have I Been Pwned*, perlu dipertimbangkan aspek skalabilitas:

Database Bocor Skala Besar: Dataset seperti HIBP mengandung lebih dari 500 juta hash bocor. Tanpa optimisasi, pencocokan satu per satu akan memerlukan waktu dan sumber daya yang besar.

Solusi Skala: Indexing terenkripsi dan struktur data khusus seperti encrypted bloom filter atau homomorphic hashed index dapat mempercepat proses pencarian tanpa mengorbankan privasi.

Pemrosesan Terdistribusi: Dengan menerapkan arsitektur serverless atau microservices berbasis cloud, proses pencocokan dapat dipecah menjadi beberapa node yang bekerja paralel.

E. Evaluasi Usability dan Praktikalitas

Dari segi end user:
Keuntungan: Pengguna tidak perlu mempercayakan password mereka kepada server pihak ketiga. Bahkan jika server diretas, tidak ada informasi yang bisa digunakan karena seluruh data disimpan dalam ciphertext.

Tantangan: Proses enkripsi dan dekripsi membutuhkan waktu dan daya komputasi yang tidak secepat metode biasa. Untuk penggunaan harian, terutama oleh masyarakat awam, ini mungkin terasa lambat jika tidak dioptimalkan.

Solusi Usability: Aplikasi dapat menyediakan dua mode: cepat (fast mode) menggunakan k-anonymity, dan aman (secure mode) menggunakan enkripsi homomorfik untuk pengguna yang lebih sadar privasi.

V. KESIMPULAN

Penelitian dan implementasi yang dilakukan dalam makalah ini menunjukkan bahwa penggabungan enkripsi homomorfik dan k-anonymity dapat secara signifikan meningkatkan keamanan dan privasi dalam proses verifikasi password bocor. Dengan menggunakan skema BFV (Brakerski/Fan-Vercauteren) dan library TenSEAL, sistem

berhasil mendemonstrasikan bahwa pencocokan hash password dapat dilakukan sepenuhnya dalam bentuk terenkripsi, tanpa pernah mengungkapkan nilai asli dari password ataupun hashnya ke pihak server.

Pendekatan ini berhasil menjawab kelemahan dari skema k-anonymity tradisional, yang masih mengungkap sebagian informasi hash kepada server dan rentan terhadap serangan brute force atau deanonymisasi jika data tambahan tersedia. Dalam implementasi yang dilakukan, proses pencocokan password dilakukan melalui operasi pengurangan homomorfik, dan hasilnya hanya dapat didekripsi oleh pihak klien. Hal ini membuktikan bahwa model ini dapat diadopsi dalam arsitektur keamanan zero-trust, di mana server tidak perlu dipercaya untuk menjaga data pengguna.

Namun, meskipun pendekatan ini unggul dalam aspek keamanan, terdapat tantangan praktis yang cukup signifikan. Salah satunya adalah masalah performa dan skalabilitas. Operasi homomorfik memerlukan waktu komputasi yang lebih tinggi dan menghasilkan ciphertext berukuran besar. Ini menjadi kendala nyata untuk diterapkan dalam skala besar seperti layanan Have I Been Pwned yang memiliki ratusan juta hash bocor. Selain itu, kompleksitas penggunaan di sisi klien juga menjadi tantangan, terutama jika pengguna menggunakan perangkat dengan daya komputasi terbatas seperti smartphone.

Secara keseluruhan, pendekatan ini sangat relevan untuk digunakan dalam sistem-sistem yang membutuhkan jaminan privasi maksimal, seperti layanan pemeriksaan kredensial untuk organisasi pemerintahan, militer, atau sistem kesehatan, di mana kebocoran informasi sangat berisiko. Penggunaan homomorphic encryption dalam konteks ini menjadi alternatif kriptografi modern yang tidak hanya melindungi data saat disimpan, tetapi juga saat diproses.

VI. SARAN

Berdasarkan hasil yang telah dicapai, beberapa saran yang dapat diberikan untuk pengembangan dan penelitian lanjutan adalah:

1. Optimasi Performa dan Batching

Perlu dilakukan eksplorasi lebih lanjut terhadap teknik batching dan ciphertext packing, agar proses pencocokan terhadap banyak hash bocor dapat dilakukan dalam satu operasi homomorfik. Ini akan mengurangi waktu eksekusi dan penggunaan sumber daya secara signifikan.

2. Integrasi Hybrid

Pendekatan hybrid antara k-anonymity dan enkripsi homomorfik dapat diterapkan, di mana pengguna dapat memilih mode privasi tinggi atau mode cepat tergantung kebutuhan. Mode homomorfik dapat dijadikan sebagai fitur opsional pada layanan umum seperti HIBP.

3. Skalabilitas Cloud dan Pemrosesan Terdistribusi

Untuk mengatasi keterbatasan lokal, server-side processing dapat dikembangkan dengan sistem paralel dan terdistribusi berbasis cloud untuk menangani permintaan dalam skala besar, tanpa mengurangi privasi pengguna.

4. Penerapan Index Terenkripsi

Penggunaan struktur data terenkripsi seperti homomorphic bloom filter atau private information retrieval dapat mempercepat proses pencarian dalam database hash bocor tanpa membuka isi database tersebut.

5. Studi Komparatif dan Standardisasi

Perlu dilakukan studi komparatif terhadap berbagai skema enkripsi homomorfik seperti CKKS dan BGV untuk menemukan skema yang paling efisien dalam konteks pengecekan password bocor. Selain itu, pengembangan standar protokol privasi untuk layanan pengecekan kredensial juga penting untuk mendukung adopsi luas.

6. Edukasi dan Inklusivitas Pengguna

Mengingat kompleksitas kriptografi homomorfik, penting untuk mengedukasi pengguna umum mengenai manfaat privasi dari pendekatan ini. Antarmuka pengguna yang sederhana dan pengalaman pengguna yang baik akan meningkatkan adopsi sistem secara luas.

7. Evaluasi Keamanan Lanjutan

Simulasi dan analisis terhadap potensi serangan baru, seperti serangan berbasis noise atau side-channel pada library homomorfik yang digunakan, perlu dilakukan untuk memastikan sistem tetap tangguh dalam jangka panjang.

Dengan pengembangan yang tepat, pendekatan enkripsi homomorfik yang dikombinasikan dengan prinsip k-anonymity memiliki potensi besar untuk menjadi solusi masa depan dalam menjaga keamanan dan privasi digital pengguna, khususnya dalam era yang semakin rentan terhadap eksploitasi data pribadi.

REFERENCES

[1] [1]L. SWEENEY, “k-ANONYMITY: A MODEL FOR PROTECTING PRIVACY,” International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, vol. 10, no. 05, pp. 557–570, Oct. 2002, doi: https://doi.org/10.1142/s0218488502001648.

[2] [2]“Have I Been Pwned: API v3,” Haveibeenpwned.com, 2022. https://haveibeenpwned.com/API/v3#PwnedPasswords

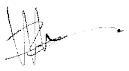
[3] [3]A. Godoy, “Introduction to Microsoft SEAL: Exploring Homomorphic Encryption,” Medium, Feb. 03, 2025. https://medium.com/@anderson.buenogod/introduction-to-microsoft-seal-exploring-homomorphic-encryption-910f1af0cb56 (accessed Jun. 15, 2025).Transl. J. Magn. Japan, vol. 2, pp. 740-741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].

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, 20 Juni 2025

Recoverable Signature

X 

Signed by: ab8abf11-c5d8-4fb8-8b82-e92580dd306b

Wisyendra L 18222095