Jurnal Pendidikan dan Teknologi Indonesia (JPTI)
Vol. 5, No. 9, September 2025, Hal. 2648-2659
DOI: https://doi.org/10.52436/1.jpti.1026
p-ISSN: 2775-4227
e-ISSN: 2775-4219

Pengembangan dan Evaluasi Protokol VHE-PIR Berbasis Multi-Server untuk Pengambilan Informasi yang Privat dan Skalabel

Slamet Widodo[*1], Fandy Setyo Utomo[2], Berlilana[3]

[1,2,3]Program Studi Magister Ilmu Komputer, Universitas Amikom Purwokerto, Indonesia
Email: [1]swidodo009@gmail.com, [2]fandysetyoutomo@amikompurwokerto.ac.id,
[3]berlilana@amikompurwokerto.ac.id

Abstrak

Penelitian ini bertujuan untuk mengembangkan protokol Private Information Retrieval (PIR) berbasis multi-server yang mengintegrasikan enkripsi homomorfik terverifikasi (Verifiable Homomorphic Encryption - VHE) untuk meningkatkan privasi, efisiensi, dan keandalan dalam pengambilan informasi dari basis data. Protokol ini dirancang untuk mengatasi keterbatasan arsitektur server tunggal, seperti risiko kegagalan sistem, beban kerja yang tinggi, dan keterbatasan skalabilitas. Metode penelitian melibatkan distribusi basis data ke beberapa server, penggunaan public key dan private key untuk enkripsi dan verifikasi hasil, serta penerapan modul akselerasi untuk mendukung pemrosesan paralel. Simulasi dilakukan pada lingkungan terdistribusi untuk mengevaluasi waktu respons, penggunaan memori, serta kemampuan failover dalam kondisi server bermasalah. Hasil penelitian menunjukkan bahwa pada skenario normal, arsitektur multi-server secara konsisten memiliki waktu respons lebih rendah dibandingkan arsitektur server tunggal, baik untuk protokol non-VHE maupun VHE-PIR. Misalnya, pada 200 pengguna, waktu respons multi-server VHE adalah 3,6070 detik dibandingkan dengan 4,2433 detik pada single server. Selain itu, dalam kondisi server bermasalah, arsitektur multi-server tetap mampu melayani permintaan dengan mendistribusikan beban ke server lain, sementara server tunggal mengalami kegagalan total. Protokol VHE-PIR menunjukkan privasi yang lebih tinggi dengan memastikan elemen yang diakses tidak dapat diketahui oleh server, meskipun memerlukan sumber daya memori dan waktu respons sedikit lebih besar dibandingkan protokol non-VHE. Implikasi dari penelitian ini mencakup kontribusi akademik dalam desain protokol PIR tahan gangguan dan kontribusi praktis terhadap sistem informasi modern yang membutuhkan skala besar, kecepatan akses, serta jaminan kerahasiaan. Penelitian ini relevan untuk implementasi nyata, dan membuka ruang eksplorasi lebih lanjut dalam penerapan teknologi PIR di lingkungan cloud publik dan sistem basis data terdistribusi.

Kata kunci: *Data Privacy, Failover Mechanism, Multi-Server Architecture, Private Information Retrieval (PIR), Verifiable Homomorphic Encryption (VHE).*

Development and Evaluation of a Multi-Server-Based VHE-PIR Protocol for Private and Scalable Information Retrieval

Abstract

This research aims to develop a multi-server-based Private Information Retrieval (PIR) protocol integrating Verifiable Homomorphic Encryption (VHE) to enhance privacy, efficiency, and reliability in retrieving information from databases. The protocol is designed to address the limitations of single-server architecture, such as system failure risks, high workload, and scalability constraints. The research methodology involves distributing databases across multiple servers, using public and private keys for encryption and result verification, and implementing acceleration modules to support parallel processing. Simulations were conducted in a distributed environment to evaluate response time, memory usage, and failover capability under server failure conditions. The results show that, under normal scenarios, the multi-server architecture consistently achieves lower response times than the single-server architecture for both non-VHE and VHE-PIR protocols. For instance, with 200 users, the response time for the multi-server VHE-PIR is 3.6070 seconds compared to 4.2433 seconds for the single server. Furthermore, under server failure conditions, the multi-server architecture can still serve user requests by redistributing the load to remaining servers, whereas the single server experiences total failure. The VHE-PIR protocol also provides higher privacy by ensuring that the accessed elements cannot be identified by the server, although it requires slightly more memory and response time than the non-VHE protocol. The implications of this research include academic contributions in fault-tolerant PIR protocol design and practical contributions to modern information systems that require large scale, access speed, and confidentiality assurance. This research

p-ISSN: 2775-4227
e-ISSN: 2775-4219

is relevant for real-world implementation and opens space for further exploration in applying PIR technology in public cloud environments and distributed database systems.

Keywords: *Data Privacy, Failover Mechanism, Multi-Server Architecture, Private Information Retrieval (PIR), Verifiable Homomorphic Encryption (VHE).*

1. PENDAHULUAN

Pertumbuhan teknologi informasi yang pesat memungkinkan akses mudah terhadap berbagai data publik, mulai dari data pemerintah, kesehatan, pendidikan, hingga transaksi komersial. Namun, kemudahan ini beriringan dengan meningkatnya tantangan terhadap privasi data pengguna [1][2][3]. Aktivitas pengambilan data dari basis data publik memiliki risiko tinggi dalam mengungkapkan identitas, preferensi, atau kebutuhan pengguna, sehingga privasi pengguna menjadi isu krusial [4][5]. Potensi kebocoran informasi pribadi dalam aktivitas ini dapat digunakan untuk profiling, pengawasan, atau bahkan eksploitasi informasi sensitif, sehingga urgensi terhadap perlindungan privasi semakin meningkat [6]. Oleh sebab itu, dibutuhkan mekanisme yang dapat menjamin pengambilan informasi secara selektif tanpa mengorbankan privasi pengguna.

Private Information Retrieval (PIR) muncul sebagai solusi kriptografi yang memungkinkan pengguna untuk mengambil data secara rahasia tanpa memberi tahu penyedia layanan tentang data mana yang diakses [7][8]. PIR bekerja dengan cara menyusun query pengguna dalam bentuk yang terenkripsi atau tersembunyi, sehingga pemilik data hanya mengetahui bahwa suatu kueri dilakukan tanpa mengetahui data spesifik yang diambil [9]. Berbagai penelitian telah dilakukan untuk meningkatkan efektivitas PIR, salah satunya dengan pendekatan berbasis enkripsi homomorfik. Metode ini memungkinkan komputasi atau pemrosesan dilakukan langsung pada data dalam kondisi terenkripsi tanpa harus mendekripsinya terlebih dahulu, sehingga menjamin tingkat privasi yang tinggi sekaligus menjaga integritas data [10][11][12].

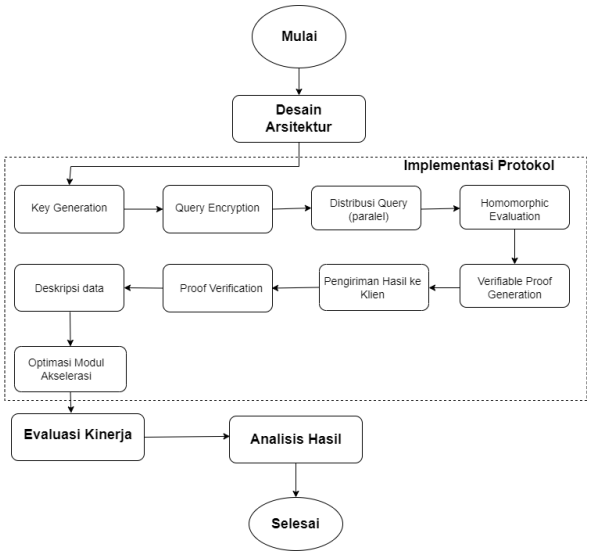
Namun demikian, sebagian besar solusi PIR yang dikembangkan masih berbasis server tunggal, yang memiliki beberapa kelemahan kritis. Kelemahan tersebut antara lain rentan terhadap kegagalan sistem tunggal atau single point of failure, risiko kelebihan beban kerja ketika jumlah pengguna meningkat drastis, serta tantangan dalam skalabilitas sistem untuk menangani jumlah data dan pengguna yang besar [13][14][15]. Berbagai riset telah mencoba mengatasi masalah ini, tetapi hasilnya belum optimal dalam kondisi nyata. Sebagai contoh, penelitian oleh [3] mengembangkan protokol PIR berbasis enkripsi homomorfik dengan performa tinggi, namun belum membahas secara eksplisit masalah toleransi terhadap kegagalan sistem atau redundansi data [16]. Penelitian lain oleh [5] yang berfokus pada PIR berbasis ring LWE telah memberikan kontribusi signifikan dalam aspek efisiensi dan keamanan, namun penelitian ini juga terbatas pada konteks single-server [17][18][19] dan tidak mempertimbangkan mekanisme failover dalam kondisi operasional nyata [20][21][22].

Dalam rangka mengatasi keterbatasan tersebut, penelitian ini mengusulkan protokol Verifiable Homomorphic Encryption-Based Private Information Retrieval (VHE-PIR) dengan arsitektur multi-server. Penelitian ini memiliki perbedaan mendasar dibandingkan penelitian sebelumnya karena secara eksplisit mengintegrasikan mekanisme failover yang efektif dalam skenario multi-server, serta melakukan evaluasi mendalam mengenai dampak penggunaan arsitektur ini terhadap privasi, efisiensi, dan skalabilitas. Selain itu, penelitian ini juga mencakup optimasi tambahan melalui pemrosesan paralel dan distribusi beban kerja yang optimal. Teknik ini secara langsung meningkatkan performa sistem, memungkinkan waktu respons lebih cepat dan stabilitas yang lebih baik dalam menghadapi peningkatan jumlah pengguna dan data secara signifikan. Oleh karena itu, protokol VHE-PIR berbasis multi-server yang diusulkan dalam penelitian ini diharapkan menjadi solusi praktis yang efektif, mampu menangani tantangan privasi dan skalabilitas dengan performa yang stabil dan andal dalam berbagai kondisi operasional modern.

2. METODE PENELITIAN

Penelitian ini menggunakan pendekatan eksperimental untuk mengembangkan dan menguji protokol Verifiable Homomorphic Encryption-Based PIR (VHE-PIR) berbasis multi-server. Tahapan penelitian meliputi desain arsitektur, implementasi protokol, evaluasi kinerja, dan analisis hasil.

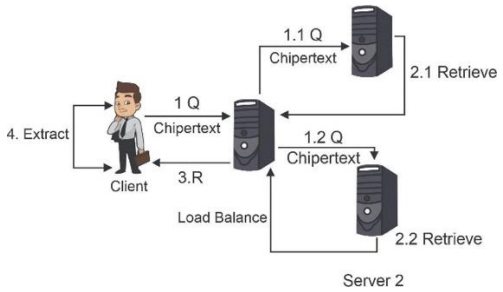
p-ISSN: 2775-4227
e-ISSN: 2775-4219



Gambar 1. Tahapan Penelitian

2.1. Desain Arstitektur VHE-PIR Multi Server

Arsitektur pada Gambar 2 adalah implementasi *Private Information Retrieval* (PIR) berbasis multi-server dengan enkripsi homomorfik dan *load balancing* untuk menjaga privasi, efisiensi, dan keandalan. Klien mengirim *query* terenkripsi yang didistribusikan oleh *load balancer* ke server, diproses secara lokal, lalu hasil terenkripsi dikirim kembali untuk dekripsi dan verifikasi. Arsitektur ini mendukung distribusi beban kerja, toleransi kegagalan, dan verifikasi data, sehingga ideal untuk privasi tinggi dan skalabilitas.



Gambar 2. Arstitektur VHE-PIR Multi Server.

Penelitian ini menggunakan dua server virtual (prosesor 4-core, RAM 8 GB, Linux) dan satu server tambahan sebagai load balancer dengan spesifikasi serupa untuk mendistribusikan beban kerja. Client dengan prosesor Intel i7, RAM 16 GB, dan sistem operasi Windows 11 digunakan untuk pengujian protokol. Jaringan lokal dengan konektivitas minimal 20 Mbps memastikan komunikasi optimal antar-server. Software yang digunakan meliputi Python untuk implementasi dan library enkripsi homomorfik seperti phe atau paillier. Bahan penelitian mencakup dataset simulasi berupa 7.000 baris data film, query terenkripsi untuk uji protokol, dan parameter kriptografi berupa kunci enkripsi dan dekripsi. Alat dan bahan ini mendukung pengembangan, implementasi, dan evaluasi sistem secara menyeluruh.

2.2. Implementasi Protokol

Dalam implementasi protokol, basis data didistribusikan ke dua server virtual dengan salinan penuh (replikasi database). Proses implementasi melibatkan pembuatan kunci global (key generation), enkripsi query pengguna dengan kunci publik, dan distribusi query secara paralel ke server. Berikut tahapan-tahapan dalam implementasi protokol:

a. Pembuatan Kunci Global (Key Generation): Menghasilkan pasangan kunci publik-privat untuk enkripsi dan dekripsi.
b. Enkripsi Query Pengguna (Query Encryption): Pengguna mengenkripsi query menggunakan kunci publik untuk menjaga kerahasiaan.

p-ISSN: 2775-4227
e-ISSN: 2775-4219

c. Distribusi Query secara Paralel ke Server: Query terenkripsi didistribusikan paralel ke beberapa server melalui load balancer.
d. Pemrosesan Query oleh Server (Homomorphic Evaluation): Server memproses query secara homomorfik tanpa membuka data asli.
e. Penghasilan Bukti Keabsahan (Verifiable Proof Generation): Server menghasilkan bukti untuk memastikan integritas hasil.
f. Pengiriman Hasil dan Bukti ke Klien: Server mengirim hasil terenkripsi bersama bukti ke klien.
g. Verifikasi oleh Klien (Proof Verification): Klien memverifikasi bukti untuk memastikan hasil valid sebelum dekripsi.
h. Dekripsi Data: Klien mendekripsi hasil akhir menggunakan kunci privat.
i. Optimasi dengan Modul Akselerasi: Menggunakan teknik paralelisme dan load balancing untuk meningkatkan efisiensi dan stabilitas system.

2.3. Evaluasi Kinerja

Evaluasi kinerja sistem dilakukan dengan fokus utama pada tiga aspek, yaitu evaluasi paralelisme, simulasi kegagalan server, dan pengukuran penggunaan memori pada server. Tujuannya adalah untuk memastikan bahwa sistem yang dikembangkan mampu bekerja secara efisien dan andal dalam berbagai kondisi penggunaan.

Evaluasi paralelisme bertujuan untuk mengukur kemampuan sistem dalam menangani banyak pengguna secara bersamaan. Dalam pengujian ini, waktu respons dicatat saat sistem melayani berbagai skenario jumlah pengguna, mulai dari 1 pengguna hingga 200 pengguna secara paralel. Selain itu, variasi ukuran elemen data yang diakses juga diuji, mulai dari 1 baris data hingga 7.000 baris data. Pengukuran waktu respons ini penting untuk melihat bagaimana sistem dapat mempertahankan kecepatan dan stabilitas ketika beban kerja meningkat, serta bagaimana efisiensi pengolahan data berubah dengan jumlah data yang berbeda.

Karena sistem menggunakan arsitektur multi-server, pengujian juga mencakup simulasi kegagalan salah satu server untuk menguji mekanisme failover yang diterapkan. Failover adalah proses dimana beban kerja yang semula ditangani oleh server yang gagal akan dialihkan secara otomatis ke server lain agar layanan tetap berjalan tanpa gangguan. Efektivitas failover diukur dengan mengamati seberapa cepat dan mulus sistem beralih ke server cadangan, serta apakah terjadi gangguan signifikan pada waktu respons selama dan setelah kegagalan terjadi.

Evaluasi juga dilakukan untuk memantau konsumsi memori server dalam dua kondisi berbeda, yaitu kondisi normal saat sistem berjalan tanpa gangguan, dan kondisi bermasalah saat terjadi beban tinggi atau kegagalan server. Pengukuran penggunaan memori ini penting untuk menilai efisiensi sumber daya sistem dan mengidentifikasi potensi kebocoran memori atau bottleneck yang dapat mempengaruhi performa secara keseluruhan.

2.4. Analisis Hasil

Hasil pengujian dianalisis untuk membandingkan performa waktu respons antara arsitektur multi-server dan server tunggal, mengevaluasi keandalan sistem dalam kondisi server bermasalah, serta menilai dampak enkripsi homomorfik terhadap efisiensi waktu respons dan penggunaan memori. Pendekatan ini memastikan bahwa sistem yang dikembangkan dapat memenuhi kebutuhan privasi, efisiensi, dan keandalan secara menyeluruh.

3. HASIL DAN PEMBAHASAN

3.1. Pengembangan protokol.

Dalam rangka mewujudkan sistem Private Information Retrieval (PIR) yang aman, efisien, dan andal, terutama dalam arsitektur multi-server, protokol Verifiable Homomorphic Encryption-Based PIR (VHE-PIR) dikembangkan dengan serangkaian tahapan kriptografi dan mekanisme verifikasi. Protokol ini dirancang untuk menjaga privasi query pengguna sekaligus memastikan keabsahan hasil pengambilan data melalui proses enkripsi, evaluasi homomorfik, serta bukti kriptografi yang dapat diverifikasi. Berikut ini penjabaran detail tahapan utama dalam sistem VHE-PIR:

3.1.1. Pembuatan Kunci Global (Key Generation)

Tahapan ini adalah proses awal dalam protokol VHE-PIR, di mana sistem menghasilkan pasangan kunci publik dan kunci privat. Kunci publik digunakan untuk mengenkripsi query pengguna, sementara kunci privat digunakan untuk mendekripsi hasil akhir. Proses ini biasanya menggunakan skema kriptografi seperti Paillier. Berikut merupakan persamaan matematis dalam pembuatan *private key* dan *public key*:

p-ISSN: 2775-4227
e-ISSN: 2775-4219

$$(pk, sk) \leftarrow \text{KeyGen}(\lambda) \quad (1)$$

di mana λ adalah parameter keamanan yang menentukan kekuatan kriptografi yang digunakan dalam protokol [3][5]. Dalam penelitian ini proses tersebut diterapkan dalam script fungsi untuk pembuatan *private key* dan *public key*:

```
# Generate Paillier keys
def generate_paillier_keys():
    public_key, private_key = paillier.generate_paillier_keypair()
    return public_key, private_key
```

3.1.2. Enkripsi Query Pengguna (Query Encryption)

Pada tahap ini, pengguna mengenkripsi permintaan informasi (q) menggunakan kunci publik pk yang telah dibuat pada tahap sebelumnya. Enkripsi ini bertujuan menjaga kerahasiaan informasi yang diminta pengguna sehingga penyedia layanan tidak mengetahui data spesifik yang diakses oleh pengguna. Proses enkripsi dijelaskan dalam persamaan berikut:

$$c = \text{Enc}(pk, q) \quad (2)$$

dengan c sebagai ciphertext query terenkripsi [3] [4]. Berikut script fungsi untuk proses enkripsi:

```
# Encrypt integer parameter
def encrypt_parameter(public_key, value):
    return public_key.encrypt(value)

# Simulasi parameter terenkripsi
start_time = time.time()
encrypted_param = encrypt_parameter(public_key, 10)
print(f"Encrypted Parameter: {encrypted_param.ciphertext()}")

# Ganti parameter dengan yang terenkripsi
query = query.format(decrypted_param)
```

3.1.3. Distribusi Query secara Paralel ke Server

Setelah query terenkripsi (c) dibuat, query ini didistribusikan secara paralel melalui mekanisme *load balancing* ke beberapa server. Distribusi ini bertujuan untuk meningkatkan efisiensi pemrosesan query, memastikan toleransi terhadap kegagalan server, serta mengurangi beban kerja per server.

$$c_1 = c, \quad c_2 = c \quad (3)$$

dengan *c1* adalah query terenkrisi pada server 1, dan *c2* adalah query terenkripsi pada server 2. Langkah ini dilakukan untuk mendukung redundansi dan toleransi terhadap kegagalan server [10][12].

3.1.4. Pemrosesan Query oleh Server (Homomorphic Evaluation)

Setiap server menerima query terenkripsi dan memprosesnya secara homomorfik. Proses homomorphic evaluation memungkinkan server untuk melakukan operasi perhitungan langsung terhadap data yang terenkripsi tanpa perlu membuka atau mendekripsi data tersebut, sehingga privasi pengguna tetap terjamin sepanjang proses pemrosesan data. Proses ini dapat dijelaskan menggunakan persamaan berikut :

$$r_i = \text{Eval}(D_i, c_i) \quad (4)$$

Di mana Di adalah data di server ke-i, dan ri adalah hasil ciphertext komputasi query [3][4][5].

3.1.5. Penghasilan Bukti Keabsahan (Verifiable Proof Generation)

Setelah server menyelesaikan proses komputasi homomorfik, server menghasilkan bukti kriptografi yang dapat diverifikasi oleh klien. Bukti ini bertujuan untuk memastikan integritas data hasil proses, sehingga klien yakin bahwa hasil yang diterima benar-benar valid dan tidak dimanipulasi selama proses pemrosesan. Server menghasilkan bukti kriptografi πi yang membuktikan bahwa hasil tersebut valid dan tidak dimanipulasi, tanpa mengungkap data asli maupun query.

p-ISSN: 2775-4227
e-ISSN: 2775-4219

$$\pi i = \text{Prove}(D_i, c_i, r_i, sk_i) \quad (5)$$

Mekanisme ini memastikan integritas hasil dan mencegah manipulasi dari server [5][11].

3.1.6. Pengiriman Hasil dan Bukti ke Klien

Server kemudian mengirimkan hasil pemrosesan yang masih terenkripsi ri bersama bukti keabsahan πi secara bersamaan ke klien. Proses ini menjamin bahwa hasil pemrosesan dapat divalidasi keasliannya oleh klien sebelum data sebenarnya diakses.

3.1.7. Verifikasi oleh Klien (Proof Verification)

Klien menerima hasil terenkripsi beserta bukti keabsahan dari server. Pada tahap ini, klien melakukan verifikasi terhadap bukti tersebut untuk memastikan integritas dan keaslian data hasil pemrosesan sebelum mendekripsinya. Tahap ini sangat penting untuk memastikan bahwa hasil akhir yang diterima oleh klien adalah valid.

$$Verify(pk, c_i, r_i, \pi_i) \rightarrow \{\text{true,false}\} \quad (6)$$

Verifikasi ini penting untuk memastikan hasil yang diterima dapat dipercaya sebelum dekripsi dilakukan [5][11][9].

3.1.8. Dekripsi Data

Setelah bukti berhasil diverifikasi, klien mendekripsi hasil pemrosesan menggunakan kunci privat yang telah dihasilkan pada tahap Key Generation. Dekripsi ini memungkinkan klien untuk memperoleh data asli yang diinginkan secara aman tanpa kompromi terhadap privasi.

$$m = Dec(sk, r) \quad (7)$$

Di mana r adalah hasil gabungan dari server [3][11]. Berikut script fungsi untuk proses deskripsi:

```
# Decrypt integer parameter
def decrypt_parameter(private_key, encrypted_value):
    return private_key.decrypt(encrypted_value)
```

hasil output dari mulai proses key generation sampai proses deskripsi adalah sebagai berikut:

```
Generating Paillier keys...
Simulating multiple users with encrypted queries...
Encrypted Parameter: 57517274404909130909191970500147916323803312945152384834102006329879278471383055374154628488831026841934090572679393301429502160397165503530279061251987351606421
Decrypted Parameter: 10
Query executed in 1.0109 seconds.
Hasil Query:
('A Perfect Crime', 2020)
('A 2nd Chance', 2011)
('A 3 Minute Hug', 2019)
("A Babysitter's Guide to Monster Hunting", 2020)
('A Bad Moms Christmas', 2017)
('A Beautiful Life', 2011)
('A Billion Colour Story', 2016)
('A Boy Called Po', 2016)
('A Boy Name Flora A', 2017)
('A Bridge Too Far', 1977)
('A California Christmas', 2020)
('A Champion Heart', 2018)
('A Chaster Marriage', 2016)
('A Choo', 2020)
('A Christmas Catch', 2018)
('A Christmas Prince', 2017)
('A Christmas Prince: The Royal Baby', 2019)
('A Christmas Prince: The Royal Wedding', 2018)
('A Christmas Special: Miraculous: Tales of Ladybug & Cat Noir', 2016)
('A Cinderella Story', 2004)
('A Cinderella Story: Christmas Wish', 2019)
('A Clockwork Orange', 1971)
('A Dangerous Woman', 1993)
('A Fairly Odd Summer', 2014)
```

Gambar 3 Hasil output proses tahapan VHE-PIR

3.1.9. Optimasi dengan Modul Akselerasi

Tahap optimasi menggunakan dua Teknik yaitu pemrosesan parallel dan load balancing. Pemrosesan Paralel: Beban kerja query dibagi dan dikerjakan bersamaan oleh kedua server, sehingga mempercepat waktu respons dan meningkatkan efisiensi [9]. Load Balancing: Beban kerja didistribusikan optimal agar tidak terjadi bottleneck pada salah satu server, menjaga kestabilan sistem terutama saat jumlah pengguna besar [10][8].

p-ISSN: 2775-4227
e-ISSN: 2775-4219

3.2. Evaluasi Kinerja

Penelitian ini dilakukan melalui serangkaian percobaan yang dirancang untuk mengevaluasi kinerja protokol Verifiable Homomorphic Encryption-Based PIR (VHE-PIR) dalam arsitektur multi-server. Hasil dari pengujian sebelumnya yang dilakukan oleh [3] dalam pengukuran waktu satu fase pengambilan 1 baris data (single server) didapatkan hasil protokol BV-PIR yang membutuhkan rata-rata waktu eksekusi selama 24 detik, VSPIR 27,08 detik. Sebaliknya, VHE-PIR hanya membutuhkan waktu 3,86 detik, menjadikannya yang tercepat di antara semua protokol yang diuji. Bahkan ketika modul percepatan (Acceleration Module, AM) dinonaktifkan, VHE-PIR tanpa AM tetap lebih efisien dengan waktu eksekusi 4,50 detik. Penelitian ini tidak membandingkan BV-PIR dan VSPIR dalam arsitektur multi server, melainkan hanya berfokus kepada pengujian VHE-PIR dan tanpa penggunaan enkripsi dalam arsitektur multi server dengan kondisi dataset dan spesifikasi disesuaikan dengan penelitian sebelumnya.

Pengujian diawali dengan pengaturan dua server virtual dengan spesifikasi 4-core prosesor, RAM 8 GB, serta satu server tambahan sebagai load balancer. Dataset simulasi berupa 7.000 baris data film diunggah ke setiap server dalam bentuk replikasi penuh. Percobaan pertama dilakukan untuk mengevaluasi waktu respons berdasarkan jumlah pengguna yang bervariasi (1, 10, 100, hingga 200 pengguna) dengan memanfaatkan load balancing pada multi-server untuk mendistribusikan beban kerja. Selanjutnya, pengujian dilakukan berdasarkan jumlah data yang diambil (1, 500, 2.000, hingga 7.000 data) untuk menilai pengaruh ukuran query terhadap waktu respons pada protokol non-VHE dan VHE-PIR. Dalam skenario kegagalan server, salah satu server dinonaktifkan secara sengaja untuk menguji efektivitas mekanisme failover, dengan query didistribusikan ulang ke server yang tersisa. Pada setiap skenario, penggunaan memori diukur baik dalam kondisi normal maupun saat terjadi kegagalan server untuk kedua protokol tersebut. Klien juga memverifikasi hasil yang diterima untuk memastikan integritas data, terutama dalam lingkungan yang tidak sepenuhnya terpercaya.

Hasil percobaan menunjukkan bahwa pada skenario normal tanpa server bermasalah, multi-server menunjukkan waktu respons yang lebih rendah dibandingkan single server untuk protokol non-VHE maupun VHE-PIR. Sebagai contoh, untuk 200 pengguna, waktu respons multi-server VHE adalah 3,6070 detik dibandingkan dengan 4,2433 detik pada single server. Hal ini menunjukkan bahwa pada kondisi Non-VHE, waktu pengambilan data pada konfigurasi multi-server selalu sedikit lebih cepat dibandingkan single-server. Saat menggunakan VHE, waktu pengambilan data secara keseluruhan meningkat pada kedua konfigurasi karena adanya overhead kriptografi, namun konfigurasi multi-server masih mempertahankan performa yang lebih baik dibandingkan single-server, terutama saat jumlah pengguna meningkat. Perbedaan performa antara single-server dan multi-server menjadi semakin signifikan seiring bertambahnya jumlah pengguna, di mana multi-server mampu menangani beban lebih efisien. Dengan demikian, dapat disimpulkan bahwa konfigurasi multi-server lebih unggul dalam efisiensi waktu pengambilan data, baik pada kondisi tanpa maupun dengan penerapan VHE, terutama saat skala pengguna bertambah banyak.

Tabel 1. Perbandingan rata - rata waktu satu fase pengambilan data berdasarkan jumlah pengguna

Komponen	Single-server Non-VHE	Multi-server Non-VHE	Single-server dengan VHE	Multi-server dengan VHE
1 user	1,7721 dtk	1,7603 dtk	3,86 dtk [3]	2,5084 dtk
10 user	1,8263 dtk	1,8157 dtk	3,7916 dtk	3,1525 dtk
100 user	3,0589 dtk	2,8756 dtk	4,2721 dtk	4,2682 dtk
200 user	4,8478 dtk	4,1656 dtk	4,2433 dtk	3,6070 dtk

Dalam eksperimen tambahan yang menguji berdasarkan jumlah data yang diambil, arsitektur multi-server tetap menunjukkan waktu respons yang lebih baik dan stabil. Pada pengambilan 7.000 data, waktu respons multi-server non-VHE adalah 1,8215 detik, sementara single server non-VHE adalah 1,8125 detik. Untuk protokol VHE-PIR, multi-server mencatat waktu 1,9028 detik dibandingkan dengan 1,9012 detik pada single server.

Tabel 2. Perbandingan rata - rata waktu satu fase pengambilan data berdasarkan jumlah data yang diambil

Komponen	Single-server Non-VHE	Multi-server Non-VHE	Single-server dengan VHE	Multi-server dengan VHE
1 data	1,6921 dtk	1,6993 dtk	1,7921 dtk	1,8012 dtk
500 data	1,7063 dtk	1,6992 dtk	1,7193 dtk	1,7250 dtk
2.000 data	1,7201 dtk	1,7392 dtk	1,8252 dtk	1,8311 dtk
7.000 data	1,8125 dtk	1,8215 dtk	1,9012 dtk	1,9028 dtk

p-ISSN: 2775-4227
e-ISSN: 2775-4219

Dalam kondisi server bermasalah, multi-server mampu tetap melayani permintaan pengguna dengan mendistribusikan beban ke server yang tersisa. Sebaliknya, single server mengalami kegagalan total (diskonek). Misalnya, pada skenario 10 pengguna dalam kondisi server bermasalah, waktu respons multi-server non-VHE adalah 1,8338 detik, sementara single server tidak dapat melayani permintaan. Hasil ini juga konsisten pada eksperimen tambahan, di mana arsitektur multi-server tetap berfungsi, seperti pada pengambilan 500 data dengan waktu respons 1,6903 detik untuk non-VHE dan 1,7320 detik untuk VHE-PIR.

Tabel 3. Perbandingan rata - rata waktu satu fase pengambilan data berdasarkan jumlah pengguna dengan kondisi satu server bermasalah

Komponen	Single-serverNon-VHE	Multi-server Non-VHE	Single-server dengan VHE	Multi-server dengan VHE
1 user	diskonek	1,7744 dtk	diskonek	2,5004 dtk
10 user	diskonek	1,8338 dtk	diskonek	3,7003 dtk
100 user	diskonek	2,9175 dtk	diskonek	4,2978 dtk
200 user	diskonek	3,9312 dtk	diskonek	4,4075 dtk

Tabel 4. Perbandingan rata - rata waktu satu fase pengambilan data berdasarkan jumlah data yang diambil dengan kondisi satu server bermasalah

Komponen	Single-server Non-VHE	Multi-server Non-VHE	Single-server dengan VHE	Multi-server dengan VHE
1 data	diskonek	1,7021 dtk	diskonek	1,7875 dtk
500 data	diskonek	1,6903 dtk	diskonek	1,7320 dtk
2.000 data	diskonek	1,7314 dtk	diskonek	1,9254 dtk
7.000 data	diskonek	1,8234 dtk	diskonek	1,8992 dtk

Penggunaan memori dalam arsitektur multi-server lebih terdistribusi, memungkinkan setiap server menggunakan memori yang lebih rendah dibandingkan single server untuk jumlah pengguna atau data yang sama. Sebagai contoh, pada 200 pengguna, server 1 dan server 2 masing-masing menggunakan 62 MB dan 72 MB (non-VHE) dalam arsitektur multi-server, dibandingkan dengan 160 MB pada single server.

Tabel 5. Perbandingan rata - rata penggunaan memori arsitektur single server dalam satu fase pengambilan data berdasarkan jumlah pengguna

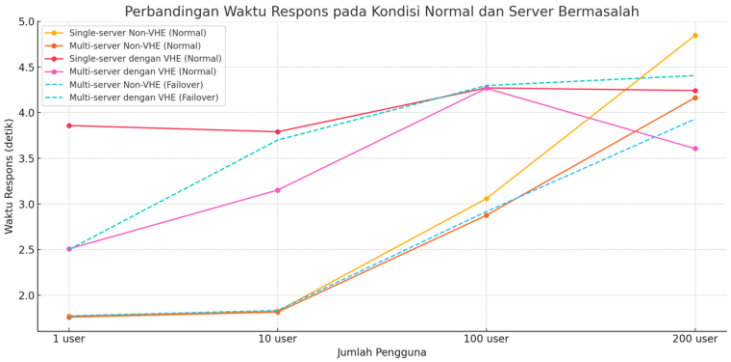
Komponen	Client	Server Non-VHE	Client	Server dengan VHE
1 user	82 Mb	26 Mb	98 Mb	27 Mb
10 user	85 Mb	33 Mb	101 Mb	31 Mb
100 user	112 Mb	91 Mb	130 Mb	92 Mb
200 user	120 Mb	160 Mb	156 Mb	158 Mb

Tabel 6. Perbandingan rata - rata penggunaan memori arsitektur multi server dalam satu fase pengambilan data berdasarkan jumlah pengguna

Komponen	Client	Server1 Non-VHE	Server 2 Non-VHE	Client	Server1 dengan VHE	Server 2 dengan VHE
1 user	80 Mb	27 Mb	25 Mb	97 Mb	28 Mb	25 Mb
10 user	84 Mb	28 Mb	27 Mb	109 Mb	27 Mb	26 Mb
100 user	120 Mb	51 Mb	49 Mb	134 Mb	50 Mb	51 Mb
200 user	145 Mb	62 Mb	72 Mb	171 Mb	68 Mb	69 Mb

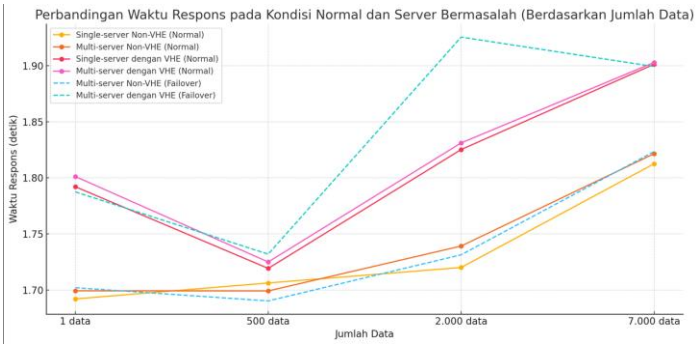
Berikut adalah grafik perbandingan berdasarkan data hasil evaluasi kinerja yang telah disajikan dalam dua kondisi, yaitu kondisi normal dan saat terjadi kegagalan server (*failover*).

p-ISSN: 2775-4227
e-ISSN: 2775-4219



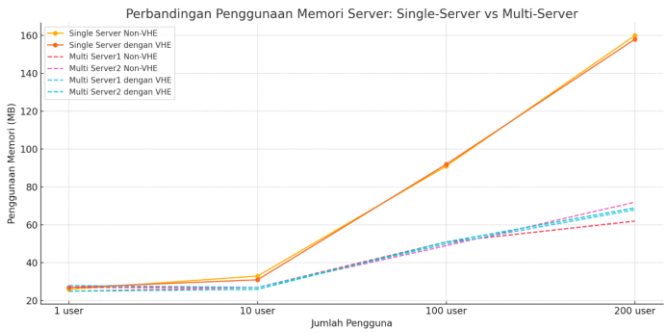
Gambar 4. Perbandingan waktu respons protokol PIR dalam kondisi normal dan saat terjadi gangguan server (failover) berdasarkan jumlah pengguna.

Grafik pada Gambar 4 menunjukkan bahwa arsitektur multi-server, baik Non-VHE maupun dengan VHE, secara konsisten memiliki performa lebih efisien dan stabil dibandingkan arsitektur single-server. Pada kondisi normal, multi-server dengan VHE mampu mengurangi waktu respons hingga lebih dari 0,6 detik pada 200 pengguna dibandingkan single-server dengan VHE. Dalam kondisi server bermasalah, arsitektur single-server mengalami kegagalan total (diskonek), sementara arsitektur multi-server tetap dapat melayani permintaan berkat implementasi mekanisme failover yang efektif. Hal ini menegaskan keunggulan multi-server dalam hal ketahanan layanan (reliability), distribusi beban kerja (load balancing), dan skalabilitas terhadap lonjakan jumlah pengguna.



Gambar 5. Perbandingan waktu respons protokol PIR pada kondisi normal dan saat terjadi kegagalan server (failover) berdasarkan jumlah data yang diambil.

Grafik pada Gambar 5 memperlihatkan bahwa arsitektur multi-server, baik dengan maupun tanpa VHE, mampu mempertahankan waktu respons yang stabil meskipun ukuran data meningkat hingga 7.000 baris. Sementara arsitektur single-server mengalami kegagalan total dalam skenario failover, sistem multi-server tetap dapat beroperasi dengan waktu respons hanya sedikit lebih tinggi dari kondisi normal. Hal ini membuktikan bahwa protokol multi-server tidak hanya lebih efisien dalam pemrosesan data besar, tetapi juga memiliki keunggulan dalam ketersediaan layanan (availability) dan kemampuan mengelola beban kerja secara terdistribusi saat terjadi gangguan.



Gambar 6. Perbandingan penggunaan memori pada sisi server antara arsitektur single-server dan multi-server dalam menangani jumlah pengguna.

p-ISSN: 2775-4227
e-ISSN: 2775-4219

Grafik pada Gambar 6 menunjukkan bahwa arsitektur single-server mengalami lonjakan penggunaan memori yang signifikan, khususnya pada 200 pengguna, mencapai lebih dari 158 MB. Sebaliknya, pada arsitektur multi-server, beban memori terdistribusi secara seimbang antara dua server, masing-masing hanya menggunakan sekitar 60–70 MB pada skenario pengguna maksimum. Distribusi ini membuktikan efisiensi arsitektur multi-server dalam pengelolaan sumber daya dan skalabilitas sistem, khususnya dalam skenario penggunaan masif.

3.3. Analisi Hasil

Secara umum, konfigurasi multi-server secara konsisten memberikan waktu respons yang lebih baik dibandingkan konfigurasi single-server dalam berbagai kondisi pengujian, baik menggunakan protokol VHE maupun non-VHE. Hal ini menunjukkan bahwa distribusi beban kerja antar server dapat secara signifikan mengurangi latensi sistem, serta memungkinkan penanganan permintaan secara paralel dan efisien.

Pada skenario peningkatan jumlah pengguna, multi-server menunjukkan performa waktu respons yang lebih stabil dan efisien. Sebagai contoh, pada 200 pengguna, konfigurasi multi-server dengan VHE mencatat waktu respons sebesar 3,6070 detik, lebih cepat dibandingkan single-server dengan VHE yang mencapai 4,2433 detik. Visualisasi perbandingan ini disajikan pada Gambar 4, yang memperlihatkan dengan jelas efektivitas konfigurasi multi-server dalam skenario normal maupun saat terjadi kegagalan server.

Pengujian berdasarkan jumlah data juga menunjukkan hasil serupa. Waktu respons untuk pengambilan data dalam jumlah besar (misalnya 7.000 data) tetap terjaga stabil pada konfigurasi multi-server, baik Non-VHE maupun dengan VHE. Gambar 5 menggambarkan tren ini, memperlihatkan bahwa protokol multi-server tetap efisien bahkan saat terjadi gangguan server.

Selain dari segi waktu, keunggulan lain dari arsitektur multi-server adalah efisiensi penggunaan memori. Berdasarkan hasil pengujian, konsumsi memori pada konfigurasi single-server meningkat drastis seiring bertambahnya jumlah pengguna, hingga mencapai 160 MB pada 200 pengguna. Sebaliknya, pada konfigurasi multi-server, beban memori didistribusikan ke dua server, masing-masing hanya menggunakan sekitar 62 MB dan 72 MB. Perbandingan ini dapat dilihat pada Gambar 6, yang menunjukkan skalabilitas arsitektur multi-server secara visual.

Tidak hanya itu, protokol multi-server juga efektif dalam menghadapi kondisi gangguan. Saat salah satu server dinonaktifkan, sistem tetap dapat berfungsi dengan mendistribusikan ulang beban ke server aktif. Mekanisme failover ini menjamin kontinuitas layanan tanpa penurunan performa signifikan.

Dengan demikian, protokol multi-server VHE-PIR menawarkan beberapa keunggulan utama dibandingkan single-server:

- Efisiensi operasional: Protokol multi-server dapat secara efektif mengurangi waktu respons dalam skenario jumlah pengguna dan data yang besar.
- Ketahanan sistem: Protokol mampu menjaga kontinuitas layanan bahkan saat salah satu server mengalami kegagalan, menunjukkan efisiensi mekanisme failover.
- Privasi yang kuat: Walaupun ada overhead tambahan karena enkripsi VHE, manfaat dalam menjaga privasi pengguna jauh lebih besar dibandingkan dengan beban tambahan tersebut.

3.4. Diskusi

Penelitian oleh Mughees (2023) [1] tentang PIR berbasis batch mencatat waktu respons rata-rata jauh lebih tinggi dibandingkan hasil yang didapat dalam penelitian ini, menegaskan bahwa VHE-PIR multi-server lebih efisien dalam skenario padat pengguna. Mughees menyoroti pentingnya efisiensi waktu respons dalam lingkungan pengguna yang padat, namun tidak mempertimbangkan aspek failover yang sangat krusial dalam skenario praktis.

Xu (2021) [4] mengusulkan protokol PIR dengan komputasi polinomial univariat yang juga menunjukkan performa yang baik, tetapi belum mencakup aspek failover. Hasil penelitian ini memperkuat temuan Xu dengan menambahkan keunggulan sistem dalam kondisi server bermasalah yang lebih realistis, menegaskan kebutuhan akan mekanisme redundansi dalam protokol PIR untuk menjamin kontinuitas layanan.

Sementara itu, penelitian oleh Ben-David (2022) [11] menekankan pentingnya verifikasi pada protokol PIR untuk memastikan integritas data. Penelitian ini secara praktis mengimplementasikan prinsip verifikasi tersebut dalam konfigurasi multi-server, memberikan hasil eksperimen yang konkret tentang efisiensi dan keandalan mekanisme verifikasi dalam skenario yang lebih kompleks dan realistis.

Grafik visualisasi waktu respons terhadap jumlah pengguna dan data secara eksplisit memperlihatkan tren peningkatan efisiensi protokol multi-server VHE-PIR dibandingkan dengan protokol-protokol yang telah dikembangkan sebelumnya, sekaligus memperjelas kontribusi signifikan penelitian ini dalam konteks praktis. Dengan visualisasi tersebut, terlihat jelas bahwa VHE-PIR multi-server mampu menjaga performa yang stabil bahkan ketika menghadapi peningkatan beban kerja yang signifikan.

p-ISSN: 2775-4227
e-ISSN: 2775-4219

Dengan demikian, penelitian ini tidak hanya membuktikan keunggulan arsitektur multi-server dengan VHE tetapi juga memberikan landasan kuat untuk implementasi nyata yang efisien dan andal dalam sistem informasi modern.

4. KESIMPULAN

Berdasarkan hasil penelitian dan eksperimen yang telah dilakukan, dapat disimpulkan bahwa pengembangan protokol Private Information Retrieval (PIR) berbasis arsitektur multi-server dengan dukungan Verifiable Homomorphic Encryption (VHE-PIR) berhasil mengatasi tantangan utama dalam pengambilan informasi dari basis data publik. Penelitian ini menunjukkan bahwa arsitektur multi-server secara signifikan lebih unggul dibandingkan arsitektur single server dalam hal efisiensi, keandalan, dan skalabilitas.

Protokol multi-server mampu memberikan waktu respons yang lebih rendah dan stabil, bahkan dalam skenario dengan jumlah pengguna atau volume data yang besar. Selain itu, arsitektur multi-server tetap beroperasi dengan baik dalam kondisi kegagalan salah satu server, berkat mekanisme failover yang mendistribusikan beban ke server yang tersisa. Sebaliknya, single server mengalami kegagalan total (diskonek) pada kondisi serupa. Penggunaan memori dalam arsitektur multi-server juga lebih efisien karena beban didistribusikan di antara beberapa server, memungkinkan setiap server untuk menggunakan sumber daya yang lebih rendah dibandingkan single server.

Protokol VHE-PIR yang diterapkan dalam sistem ini memberikan jaminan privasi tinggi dengan memastikan bahwa elemen yang diakses pengguna tidak dapat diketahui oleh server. Meski demikian, implementasi protokol ini memerlukan waktu respons dan resource memori yang sedikit lebih besar dibandingkan protokol non-VHE. Namun, kompromi ini dinilai wajar mengingat peningkatan tingkat privasi yang dihasilkan.

Secara keseluruhan, penelitian ini telah membuktikan bahwa sistem multi-server dengan protokol VHE-PIR adalah solusi yang cepat, andal, dan aman untuk pengambilan informasi dari basis data, bahkan dalam lingkungan dengan kondisi server bermasalah atau tidak sepenuhnya dapat dipercaya. Sistem ini mampu memberikan kinerja yang optimal, mendukung privasi tinggi, dan menjaga ketersediaan data dalam berbagai skenario, menjadikannya solusi ideal untuk kebutuhan pengambilan informasi modern.

Implikasi dari penelitian ini bersifat ganda. Secara akademik, penelitian ini memberikan kontribusi pada pengembangan protokol PIR modern yang tidak hanya fokus pada keamanan, tetapi juga mengintegrasikan efisiensi dan ketahanan sistem dalam skenario operasional nyata. Secara praktis, temuan ini membuka peluang bagi pengembangan sistem informasi yang berskala besar, seperti sistem layanan publik, penyimpanan cloud, atau basis data terdistribusi, untuk mengadopsi arsitektur multi-server dengan pendekatan VHE demi menjaga privasi dan kontinuitas layanan.

Penelitian ini hanya membahas kegagalan satu server dalam sistem multi-server. Oleh karena itu, penelitian selanjutnya direkomendasikan untuk memperluas cakupan ke skenario kegagalan multiple server atau kegagalan jaringan, yang lebih kompleks dan mencerminkan situasi dunia nyata. Selain itu, integrasi sistem dengan lingkungan produksi nyata, seperti platform cloud terdistribusi atau sistem basis data skala besar, juga penting untuk mengevaluasi kinerja dan efisiensi protokol dalam skenario non-simulatif. Pengembangan mekanisme failover adaptif serta optimalisasi pemrosesan kriptografi juga dapat menjadi arah penelitian lanjutan guna meningkatkan ketahanan dan efisiensi sistem lebih lanjut.

DAFTAR PUSTAKA

[1] M. H. Mughees, “Vectorized Batch Private Information Retrieval,” Proc. - IEEE Symp. Secur. Priv., vol. 2023, pp. 437–452, 2023, doi: 10.1109/SP46215.2023.10179329.

[2] P. Ke, “Two-Server Private Information Retrieval with Result Verification,” IEEE Int. Symp. Inf. Theory - Proc., vol. 2022, pp. 408–413, 2022, doi: 10.1109/ISIT50566.2022.9834706.

[3] F. Zhou, “Efficient private information retrievals for single-server based on verifiable homomorphic encryption,” Comput. Stand. Interfaces, vol. 93, 2025, doi: 10.1016/j.csi.2024.103961.

[4] W. Xu, “Efficient Private Information Retrieval Protocol with Homomorphically Computing Univariate Polynomials,” Secur. Commun. Networks, vol. 2021, 2021, doi: 10.1155/2021/5553256.

[5] W. K. Lin, “Doubly Efficient Private Information Retrieval and Fully Homomorphic RAM Computation from Ring LWE,” Proc. Annu. ACM Symp. Theory Comput., pp. 595–608, 2023, doi: 10.1145/3564246.3585175.

[6] J. Sun, “Efficient private information retrievals for IoT data based on bucket tree,” Comput. Electr. Eng., vol. 119, 2024, doi: 10.1016/j.compeleceng.2024.109546.

[7] L. Zhao, “Verifiable single-server private information retrieval from LWE with binary errors,” Inf. Sci. (Ny)., vol. 546, pp. 897–923, 2021, doi: 10.1016/j.ins.2020.08.071.

[8] Y. Lu, “On Single Server Private Information Retrieval With Private Coded Side Information,” IEEE

p-ISSN: 2775-4227
e-ISSN: 2775-4219

Trans. Inf. Theory, vol. 69, no. 5, pp. 3263–3284, 2023, doi: 10.1109/TIT.2023.3253078.
[9] C. Huang, "Two-Server Private Information Retrieval with High Throughput and Logarithmic Bandwidth," 2023 IEEE/CIC Int. Conf. Commun. China, ICCC 2023, 2023, doi: 10.1109/ICCC57788.2023.10233605.
[10] H. Y. Lin, "Multi-Server Weakly-Private Information Retrieval," IEEE Trans. Inf. Theory, vol. 68, no. 2, pp. 1197–1219, 2022, doi: 10.1109/TIT.2021.3126865.
[11] S. Ben-David, "Verifiable Private Information Retrieval," Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 13749, pp. 3–32, 2022, doi: 10.1007/978-3-031-22368-6_1.
[12] J. Singh, "Information-Theoretic Multi-server Private Information Retrieval with Client Preprocessing," Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 15367, pp. 423–450, 2025, doi: 10.1007/978-3-031-78023-3_14.
[13] S. Kruglik, "Two-Server Private Information Retrieval with Optimized Download Rate and Result Verification," IEEE Int. Symp. Inf. Theory - Proc., vol. 2023, pp. 1354–1359, 2023, doi: 10.1109/ISIT54713.2023.10206529.
[14] W. Zhang, "VPIR: an efficient verifiable private information retrieval scheme resisting malicious cloud server," Telecommun. Syst., vol. 86, no. 4, pp. 743–755, 2024, doi: 10.1007/s11235-024-01162-1.
[15] H. Y. Lin, "The capacity of single-server weakly-private information retrieval," IEEE J. Sel. Areas Inf. Theory, vol. 2, no. 1, pp. 415–427, 2021, doi: 10.1109/JSAIT.2021.3056327.
[16] A. Heidarzadeh, "The Linear Capacity of Single-Server Individually-Private Information Retrieval With Side Information," IEEE Int. Symp. Inf. Theory - Proc., vol. 2022, pp. 2833–2838, 2022, doi: 10.1109/ISIT50566.2022.9834466.
[17] S. A. Obead, "Single-Server Pliable Private Information Retrieval With Side Information," IEEE Int. Symp. Inf. Theory - Proc., vol. 2023, pp. 1526–1531, 2023, doi: 10.1109/ISIT54713.2023.10206829.
[18] A. Heidarzadeh, "Single-Server Individually-Private Information Retrieval: A Combinatorial Approach," 2021 IEEE Inf. Theory Work. ITW 2021 - Proc., 2021, doi: 10.1109/ITW48936.2021.9611358.
[19] S. Kadhe, "Single-server private information retrieval schemes are equivalent to locally recoverable coding schemes," IEEE J. Sel. Areas Inf. Theory, vol. 2, no. 1, pp. 391–402, 2021, doi: 10.1109/JSAIT.2021.3053579.
[20] M. Zhou, "Optimal Single-Server Private Information Retrieval," Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 14004, pp. 395–425, 2023, doi: 10.1007/978-3-031-30545-0_14.
[21] G. N. Alfarano, "A survey on single server private information retrieval in a coding theory perspective," Appl. Algebr. Eng. Commun. Comput., vol. 34, no. 3, pp. 335–358, 2023, doi: 10.1007/s00200-021-00508-5.
[22] H. Corrigan-Gibbs, "Single-Server Private Information Retrieval with Sublinear Amortized Time," Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 13276, pp. 3–33, 2022, doi: 10.1007/978-3-031-07085-3_1.