Revisiting Hybrid Private Information Retrieval

Daniel Günther
guenther@encrypto.cs.tu-darmstadt.de
Technical University of Darmstadt
Darmstadt, Germany

Thomas Schneider
schneider@encrypto.cs.tu-darmstadt.de
Technical University of Darmstadt
Darmstadt, Germany

Felix Wiegand
felix-wiegand@protonmail.ch
Technical University of Darmstadt
Darmstadt, Germany

ABSTRACT

Private Information Retrieval (PIR) allows a client to request entries from a public database held by k servers without revealing any information about the requested data to the servers. PIR is classified into two classes: (i) *Multi-server PIR* protocols where the request is split among $k \geq 2$ non-colluding servers, and (ii) *Single-server PIR* protocols where exactly $k = 1$ server holds the database while the query is protected via certain computational hardness assumptions.

Devet & Goldberg (PETS '14) showed that both can be combined into one recursive PIR protocol in order to improve the communication complexity. Their *hybrid PIR* protocol is instantiated with the multi-server PIR protocol of Goldberg (S&P'07) and the single-server PIR protocol by Melchar & Gaborit (WEWoRC'07), resulting in online request runtime speedups and guaranteeing at least partial privacy if the multi-server PIR servers do in fact collude.

In this work we show that the hybrid PIR protocol by Devet & Goldberg still has practical relevance by designing a hybrid approach using the state-of-the-art multi-server protocol CIP-PIR (Günther et al., ePrint '21/823) and the single-server protocol SealPIR (Angel et al., S&P '18). Our novel hybrid PIR protocol massively improves the linear communication complexity of CIP-PIR and obtains the strong property of client-independent preprocessing, which allow batch-preprocessing among multiple clients *without* the clients being involved. We implement and benchmark our protocol and get speedups of $\approx 4.36\times$ over the original implementation of Devet & Goldberg (8 GiB DB), speedups of $\approx 26.08\times$ (8 GiB DB) over CIP-PIR, and speedups of $\approx 11.16\times$ over SealPIR (1 GiB DB).

CCS CONCEPTS

• **Security and privacy** → **Privacy-preserving protocols**; • **Computing methodologies** → **Distributed algorithms**; • **Information systems** → *Information retrieval query processing*.

KEYWORDS

Private Information Retrieval, Large-Scale Applications

ACM Reference Format:
Daniel Günther, Thomas Schneider, and Felix Wiegand. 2021. Revisiting Hybrid Private Information Retrieval. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security (CCS '21), November 15–19, 2021, Virtual Event, Republic of Korea.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3460120.3485346

1 INTRODUCTION

Private Information Retrieval (PIR) [4] allows requesting a database element $DB[i_0]$ from a database DB consisting of n elements held by k servers without leaking i_0 or $DB[i_0]$ to the server(s). *Multi-server PIR* [5, 10, 11] provide this guarantee by splitting the client's request among $k \geq 2$ servers from which a subset of $t \leq k$ servers are assumed to be non-colluding. In contrast, *single-server PIR* protocols [2, 15] instead rely on cryptographic hardness assumptions, e.g., by utilizing homomorphic encryption. However, single-server PIR protocols are comparatively very inefficient as expensive cryptographic operations over the complete database are necessary, while the multi-server PIR protocols just need to compute very cheap operations (e.g., XOR) and thus are more efficient.

In hybrid PIR as proposed by Devet & Goldberg [7], both approaches are combined in a recursive protocol, where the multi-server PIR part selects a subset of the database on which single-server PIR is performed. This solution improves the linear communication complexity of the multi-server PIR part, limits the computational overhead of the single-server PIR protocol, and provides partial privacy in case of server collusion as only a subset of possible requested database elements is leaked. Especially in large-scale applications, where a single-server solution has too much computation overhead, and users may not trust the servers not to collude, a hybrid protocol can be a useful compromise. Since Devet & Goldberg [7] designed and evaluated their hybrid PIR protocol in 2014, more efficient PIR protocols have been published.

The multi-server protocol CIP-PIR [11] moves the majority of the server's online computation into a client-independent offline phase that can be efficiently batch-processed. The single-server protocol SealPIR [2] allows querying the database with a number of ciphertexts constant in the number of database elements, reducing the communication required.

PIR applications. Prominent PIR applications include patent database look-ups [3], anonymous messaging [6], private inquiries into browser blocklists [14], and Certificate Transparency databases [13]. Recently, PIR was used in the context of Privacy-Preserving Epidemiological Modelling (PEM) [12] as an effective measure against the COVID-19 pandemic. This framework allows performing distributed epidemiological simulations in a privacy-friendly way, using real world data collected by mobile tracking applications.

Contributions. In this work, we show that Devet & Goldberg's [7] hybrid PIR protocol from 2014 still has practical relevance by instantiating this hybrid protocol with the multi-server PIR protocol

CIP-PIR [11] and the single-server PIR protocol SealPIR [2]. We implement the resulting protocol in Rust and measure its performance for very large databases, especially with small element sizes, and compare it to Devet & Goldberg's original implementation, as well as pure CIP-PIR and SealPIR.

2 PIR PROTOCOLS

CIP-PIR [11]. CIP-PIR by Günther et al. [11] is a multi-server PIR scheme that involves k servers from which $2 \leq t \leq k$ are assumed to be non-colluding. It is the first PIR protocol that can perform a preprocessing phase completely independent of the client and thus reduces the server's online computation time by factor $k - 1/k$ over the traditional multi-server schemes [5, 6] building up on Chor et al.'s PIR [4] and PIR based on distributed point functions [9, 13].

The core idea of Chor et al.'s PIR [4] works as follows: Let us assume the client aims to retrieve the data block i_0 from a database DB of n blocks. Then, the client generates an n bit zero string q with a '1' at position i_0. The client generates $k - 1$ random n-bit strings $q_1, \ldots, q_{k-1}$ and computes the so-called flip query $q_n = q \bigoplus_{i=1}^{k-1} q_i$ and sends q_i to server i. The servers then compute the answer of the query as $a_i = \bigoplus_{j=1}^{n} q_i[j] \cdot DB[j]$ and finally, the client computes $DB[i_0] = \bigoplus_{i=1}^{k} a_i$.

RAID-PIR by Demmler et al. [5, 6] improves over Chor et al.'s PIR [4] by splitting the database into k chunks and the flip query, which cancels out all unwanted bits from the random queries, is spread among all k servers s.t. each server holds the flip query part of exactly one chunk. In addition, the random parts of the queries are derived from a seed via a pseudorandom generator PRG. Consequently, the client then only needs to send to each server the flip chunk query and a κ bit seed, where κ is the security parameter.

CIP-PIR [11] goes one step further and lets the servers choose the seeds in order to precompute the random parts of the query in an offline phase. Since there is no information from the client needed, the servers can precompute $k - 1/k$ of the answer and even can do this for many clients in parallel, which allows for massive parallelization, e.g., on a GPU, and batch processing.

Günther et al. [11] showed that CIP-PIR outperforms all state-of-the-art PIR implementations for databases up to 16 TB including PIR based on distributed point functions [9, 13], which are well-known for their logarithmic communication complexity, while CIP-PIR has linear communication complexity. However, it turned out that the online computation time is the bottleneck of multi-server PIR, which CIP-PIR massively improves.

SealPIR [2] The base for SealPIR is the single-server PIR protocol XPIR by Aguilar-Melchor et al. [1], which uses lattice-based homomorphic encryption. SealPIR by Angel et al. [2] improves over XPIR by focussing on compressing the query to reduce network costs.

At its core, SealPIR – like XPIR – is a traditional single-server PIR protocol relying on additively homomorphic encryption, where the query consists of an encrypted '1' for the desired element and an encrypted '0' for every other one. The server multiplies each query element with the corresponding database element and sums the results together. The client then simply decrypts the result to obtain the desired element.

Since a full query consisting of a ciphertext for every database element is very large, SealPIR introduces a feature called *query expansion.* In the FV cryptosystem [8] employed by SealPIR, both plain and ciphertexts are polynomials of degree N. Query expansion allows indexing N elements with a single ciphertext. To index more elements, SealPIR arranges the database as a d-dimensional hypercube, and uses d ciphertexts per query to index N^d elements.

3 OUR HYBRID PROTOCOL

Devet & Goldberg [7] proposed a hybrid PIR protocol that combines multi-server PIR with single-server PIR in order to reduce the communication complexity of PIR protocols. Their framework is very modular and can be instantiated with any multi-server and single-server PIR scheme. The database DB is split into multiple sub-databases from which the sub-database, in which the client's requested element $DB[i_0]$ is included, is retrieved via multi-server PIR. Then, the servers use a single-server PIR protocol to retrieve the concrete queried element from the sub-database.

The original protocol uses Goldberg's [10] multi-server PIR protocol and the single-server PIR protocol by Melchar & Gaborit [15]. In our hybrid protocol, we instantiate the Devet & Goldberg approach with the multi-server protocol CIP-PIR [11] and the single-server protocol SealPIR [2]. The SealPIR protocol uses very efficient query compression techniques which massively reduces the communication costs of single-server PIR.

As described in section 2, CIP-PIR ist the first multi-server PIR protocol that moves a majority of the online work to an offline phase that is completely independent of the client. We can apply this novel trick for our hybrid PIR as well and get a new PIR protocol in the *Client-Independent Preprocessing* PIR model introduced by Günther et al. [11].

We implement our hybrid protocol in Rust[1], reimplement CIP-PIR to facilitate its integration into our hybrid protocol[2], and use the existing Rust bindings for SealPIR.[3]

4 EVALUATION

Experimental Setting. We perform a variety of benchmarks to compare our new hybrid PIR protocol with Devet & Goldberg's original implementation [7], as well as pure CIP-PIR [11] and SealPIR [2]. These benchmarks were performed on Core i9-7960X CPUs, with 128 GiB of RAM, and a simulated 100 Mbit/s connection.

We measure the total online runtime – the time between a client starting the request and finishing decoding the response – for a variety of database sizes and shapes, with $k = 2$ servers. All benchmarks are averages over 50 iterations.

Online runtime for different numbers of database elements. Figure 1a shows the total online runtime for increasing numbers of small ($s = 32$ bit) elements. While our new hybrid PIR protocol performs the worst for very small numbers of elements due to the overhead of a recursive protocol and the constant SealPIR query size, it outperforms Devet & Goldberg's original construction for $n \geq 2^{20}$ elements, and all other tested protocols for $n \geq 2^{26}$.

[1]https://encrypto.de/code/HybridPIR
[2]https://encrypto.de/code/CIP-PIR-Rust
[3]https://github.com/pung-project/sealpir-rust

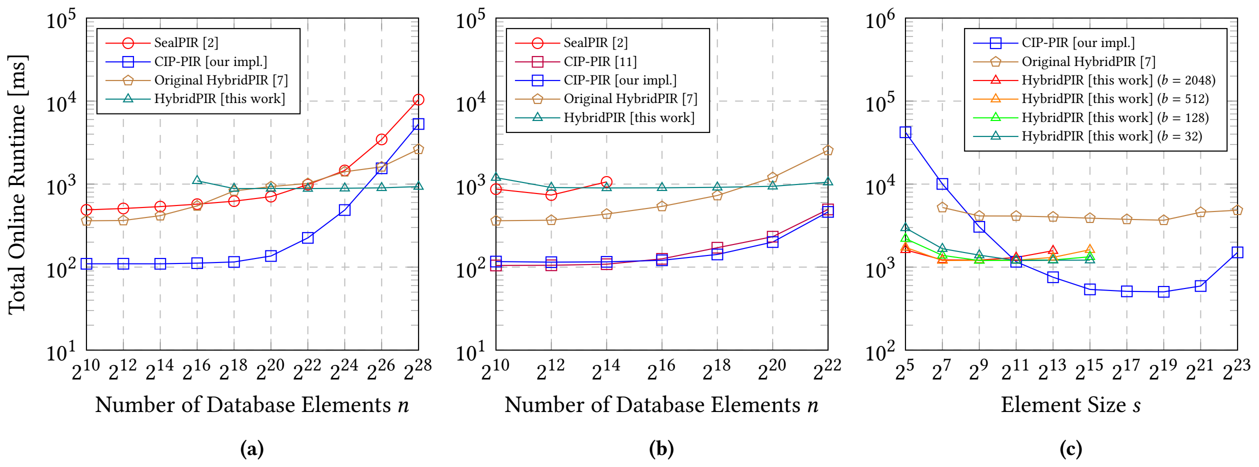

Figure 1: (a) and (b) show the total online runtime for increasing numbers of database elements for 32 and 2^{13} bit elements, respectively. (c) shows the total online runtime for differently shaped databases with a constant size of 8 GiB **($n \cdot s = 2^{36}$). For HybridPIR, different numbers of elements per block b were used.**

At $n = 2^{28}$ elements, our hybrid construction achieves speedups of $\approx 11.16\times$ over SealPIR, $\approx 5.68\times$ over CIP-PIR, and $\approx 2.82\times$ over Devet & Goldberg's hybrid protocol. While our hybrid PIR construction needs to XOR the same amount of data as pure CIP-PIR, the larger CIP-PIR blocks of our hybrid construction make up for the additional SealPIR overhead with smaller queries and better vectorization.

Figure 1b shows the total online runtime for larger ($s = 2^{13}$ bit) elements. SealPIR benchmarks could not be completed for $n > 2^{14}$ due to memory limitations. While our new hybrid construction outperforms the original one for $n \geq 2^{20}$, it fails to outperform CIP-PIR for any of the benchmark values of n. Benchmarks for more elements could not be completed due to memory limitations. Further, we see that our Rust implementation of CIP-PIR performs almost equally as well as the C++ implementation by Günther et al. [11].

Online runtime for different database shapes and block sizes. Figure 1c shows the total online runtime for differently shaped databases of size 8 GiB. Benchmarks for SealPIR could not be performed for a databases of this size. Unlike the hybrid protocols, CIP-PIR is highly sensitive to the shape of the database. Our hybrid construction achieves a speedup of $\approx 26.08\times$ over CIP-PIR for 32 bit elements, and a speedup of $\approx 4.36\times$ over the original hybrid PIR construction for 128 bit elements. The performance of CIP-PIR is optimized for element sizes super-linear in the database size as the communication complexity is minimized. We see the same behaviour in our benchmarks and conclude that our hybrid PIR outperforms CIP-PIR for databases consisting of small entries, while CIP-PIR performs better for applications with large database entries like compromised credential checking [11]. The larger CIP-PIR blocks of our hybrid construction allow us to take advantage of the best CIP-PIR performance despite smaller database elements.

ACKNOWLEDGEMENTS

This project received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (grant agreement No. 850990 PSOTI). It was co-funded by the Deutsche Forschungsgemeinschaft (DFG) — SFB 1119 CROSSING/236615297 and GRK 2050 Privacy Trust/251805230, and by the German Federal Ministry of Education and Research and the Hessen State Ministry for Higher Education, Research and the Arts within ATHENE.

REFERENCES

[1] Carlos Aguilar-Melchor, Joris Barrier, Laurent Fousse, and Marc-Olivier Killijian. 2016. XPIR : Private Information Retrieval for Everyone. In *PETs*. Springer.
[2] Sebastian Angel, Hao Chen, Kim Laine, and Srinath Setty. 2018. PIR with Compressed Queries and Amortized Query Processing. In *S&P*. IEEE.
[3] Dmitri Asonov. 2004. *Querying Databases Privately: A New Approach to Private Information Retrieval.* Springer.
[4] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan. 1995. Private Information Retrieval. In *FOCS*. IEEE.
[5] Daniel Demmler, Amir Herzberg, and Thomas Schneider. 2014. RAID-PIR: Practical Multi-Server PIR. In *CCSW*. ACM.
[6] Daniel Demmler, Marco Holz, and Thomas Schneider. 2017. OnionPIR: Effective Protection of Sensitive Metadata in Online Communication Networks. In *ACNS*. Springer.
[7] Casey Devet and Ian Goldberg. 2014. The Best of Both Worlds: Combining Information-Theoretic and Computational PIR for Communication Efficiency. In *PETs*. Springer.
[8] Junfeng Fan and Frederik Vercauteren. 2012. Somewhat Practical Fully Homomorphic Encryption. ePrint'2012/144. (2012). https://eprint.iacr.org/2012/144.
[9] Niv Gilboa and Yuval Ishai. 2014. Distributed Point Functions and Their Applications. In *EUROCRYPT*. Springer.
[10] Ian Goldberg. 2007. Improving the Robustness of Private Information Retrieval. In *S&P*. IEEE.
[11] Daniel Günther, Maurice Heymann, Benny Pinkas, and Thomas Schneider. 2021. GPU-accelerated PIR with Client-Independent Preprocessing for Large-Scale Applications. ePrint'2021/823. (2021). https://eprint.iacr.org/2021/823.
[12] Daniel Günther, Marco Holz, Benjamin Judkewitz, Helen Möllering, Benny Pinkas, and Thomas Schneider. 2020. PEM: Privacy-preserving Epidemiological Modeling. ePrint'2020/1546. (2020). https://eprint.iacr.org/2020/1546.
[13] Daniel Kales, Olamide Omolola, and Sebastian Ramacher. 2019. Revisiting User Privacy for Certificate Transparency. In *S&P*. IEEE.
[14] Dmitry Kogan and Henry Corrigan-Gibbs. 2021. Private Blocklist Lookups with Checklist. In *USENIX Security*. USENIX.
[15] Carlos Aguilar Melchor and Philippe Gaborit. 2007. A Lattice-Based Computationally-Efficient Private Information Retrieval Protocol. ePrint'2007/446. (2007). https://eprint.iacr.org/2007/446.