# Technical Report
# Secure and Private Computing
## Fuzzy PSI via Oblivious Protocol Routing

https://github.com/buehler/mcs-sec-priv-comp

**Christoph Bühler**
*University of St. Gallen*
St. Gallen, Switzerland
christoph.buehler@unisg.ch

School of Computer Science
University of St. Gallen
Switzerland
May 23, 2025

# 1 Introduction

Private Set Intersection (PSI) enables two parties, each holding a private dataset, to compute the intersection of their sets without revealing any additional information. While classical PSI protocols focus on exact matches, many practical scenarios involve noisy or imprecise data (e.g., biometric data). To address this, *fuzzy PSI* allows parties to identify approximately matching items, as determined by a public distance metric and threshold. In their 2024 work, Richardson, Rosulek, and Xu [4] introduce a novel framework for fuzzy PSI that is both flexible and efficient.

The central contribution of the paper is a general protocol construction for fuzzy PSI, which allows for arbitrary distance metrics and achieves significantly lower communication complexity compared to existing solutions. Unlike previous protocols that often depend linearly (or worse) on the similarity threshold $\delta$, the proposed protocol maintains only logarithmic dependency. This improvement is especially impactful in high-dimensional or high-threshold scenarios.

A key innovation is the use of *oblivious protocol routing*, inspired by and extending the technique introduced by Cho, Dachman-Soled, and Jarecki [2]. The authors generalize the CDJ PSI protocol – originally based on equality checks – by replacing the underlying subprotocol with a generic *private proximity test* and incorporating *conditionally-overlapping hash functions*. These hash functions ensure that approximate matches (e.g., vectors within $\ell_1$, $\ell_2$, or $\ell_\infty$ distance) are routed to a common bin, allowing the proximity test to be applied in a secure and scalable way.

The framework leverages *oblivious key-value stores* (OKVS) to encode and decode protocol messages tied to these bins, preserving privacy while supporting efficient polynomial interpolation. Additionally, the construction ensures cryptographic security in the semi-honest model and supports symmetric-key operations exclusively after an initial setup, further enhancing performance.

Richardson, Rosulek, and Xu [4] instantiate their framework for three distance metrics: $\ell_1$, $\ell_2$, and $\ell_\infty$. They use garbled circuits to implement the proximity subprotocols. Notably, for the Euclidean ($\ell_2$) case, they adopt the arithmetic garbling scheme of Ball, Malkin, and Rosulek [1], which optimizes for efficient squaring and comparisons. In terms of concrete performance, their protocol outperforms prior work such as van Baarsen and Pu [5] and Gao et al. [3] across a range of parameters, particularly when the similarity threshold is moderate to large.

Overall, the work of Richardson, Rosulek, and Xu [4] provides a highly adaptable and communication-efficient framework for fuzzy PSI, with direct applications in privacy-preserving machine learning, biometric deduplication, and other settings involving approximate matching on high-dimensional data.

The rest of this report will introduce the different components of the protocol in detail, before explaining the full protocol. The final conclusion will reflect on the implementation and the results.

# 2 Hashing Methods for Approximate Matching

A central component of the fuzzy PSI protocol by Richardson, Rosulek, and Xu [4] is the use of *conditionally-overlapping hash functions*, designed to assign similar items into at least one common bin. Listing 1 shows the implementation of the hash function $H_2$ for the $\ell_\infty$ metric that is used in the example implementation.

```rust
pub fn create_bin(val: u64, delta: u64) -> u64 {
    val / (2 * delta)
}
```

Listing 1: Creating a bin for a given value and delta in the $\ell_\infty$ metric.

Formally, for a similarity function $f(x, y) = [d(x, y) \leq \delta]$, a hash pair $(H_1, H_2)$ satisfies conditional overlap if $H_1(x) \cap H_2(y) \neq \emptyset$ whenever $f(x, y) = 1$.

### 2.1 Grid-Based Hashing for $\ell_p$ Metrics

For $\ell_p$ distances, the protocol employs a grid over $\mathbb{Z}_m^d$ with cells of side length $2\delta$. Each party maps their inputs into bins based on this grid. Two strategies are considered:

- **Asymmetric Binning:** One party (e.g., Alice) uses $H_1$ to map each input to all nearby bins within distance $\delta$, while the other party (e.g., Bob) maps each input to a single bin with $H_2$.
- **Split-Dimensional Binning:** The expansion is divided between the parties across dimensions: $H_3$ explores the first $s$ dimensions, $H_4$ the remaining $d - s$. This reduces bin overhead and balances communication costs.

### 2.2 Comparison by Metric

**$\ell_1$:** Efficient with boolean garbling; binning covers neighboring cells based on sum of absolute differences.

**$\ell_2$:** Requires squaring and summation; benefits from arithmetic garbling (e.g., [1]) for efficient proximity checks.

**$\ell_\infty$:** Based on coordinate-wise max difference; simple comparisons suffice, yielding low overhead.

### 2.3 Trade-offs

Binning affects both protocol efficiency and accuracy. Larger hash ranges increase match likelihood but incur higher communication. The framework allows tuning of binning parameters per metric to optimize for specific application needs.

## 3 Oblivious Key-Value Stores (OKVS)

To securely associate protocol messages with bin identifiers, the protocol employs *Oblivious Key-Value Stores* (OKVS). An OKVS encodes a map from keys (e.g., bin identifiers) to values (e.g., encrypted messages) in such a way that the structure of the keys remains hidden, provided the values are pseudorandom. In the fuzzy PSI protocol, each party encodes their set of bin-to-message pairs into an OKVS, which is then sent to the other party. This enables selective message decoding based on matching bins while ensuring that no additional information about the input sets or the active bins is leaked. OKVS thus serves as the cryptographic backbone for efficient, private message routing in the protocol.

### 3.1 Lagrange Polynomial OKVS

The first (attempted) implementation in this project is based on Lagrange polynomial interpolation. In this approach, the key-value pairs are interpreted as points $(x, y)$, and a polynomial of degree $n - 1$ (where $n$ is the number of pairs) is constructed such that it passes through all these points. Encoding consists of computing the coefficients of this polynomial, while decoding is simply evaluating the polynomial at the desired key. This method is information-theoretically optimal in terms of space, but encoding is computationally expensive for large $n$ due to the cost of polynomial interpolation. It is best suited for small sets or as a theoretical baseline.

### 3.2 Near-Optimal (Random Binary Matrix) OKVS

The second implementation is a near-optimal OKVS based on random binary matrices, following the "RB-OKVS" construction. Here, the key-value pairs are encoded into a binary matrix using hash functions, and the encoding process involves solving a system of linear equations over a finite field (typically using Gaussian elimination). This approach is much more efficient for large $n$, both in

terms of encoding and decoding, and is widely used in practical PSI protocols. However, it is not information-theoretically optimal and may occasionally fail to encode if the random matrix is not full rank (in which case the process is retried).

In this project, the RB-OKVS code was adapted from an older, unmaintained repository. The original code relied on outdated Rust features and compiler flags, which required significant updates to work with modern Rust toolchains. Additionally, the original implementation exhibited issues when encoding small numbers of key-value pairs. As a result, the minimal supported value for $n$ (the number of pairs) is now set to 64 in this implementation. For $n < 64$, the encoding process is likely to fail due to the properties of the underlying random matrix construction.

### 3.3 Benchmarking and Comparison

To evaluate the practical performance of both OKVS implementations, we benchmarked them using the Criterion framework in Rust. The benchmarks measure both encoding and decoding times for varying set sizes. The Lagrange-based OKVS is significantly slower for encoding as $n$ grows, while the near-optimal RB-OKVS scales much better and is suitable for large-scale applications. The results highlight the trade-off between theoretical optimality and practical efficiency, and motivate the use of RB-OKVS in real-world fuzzy PSI protocols.

As can be seen in Figure 1, the RB-OKVS implementation is significantly faster than the Lagrange polynomial implementation for both encoding and decoding. This is expected, as the Lagrange polynomial implementation is much more computationally expensive for large $n$. The mean encoding time for Lagrange polynomial with $n = 64$ is 52 ms (Figure 1a), while with RB-OKVS it is 24 $\mu$s (Figure 1b). When we reach $n = 512$, the Lagrange polynomial implementation takes 25 s (Figure 1e), while the RB-OKVS implementation takes 265 $\mu$s (Figure 1f). The decoding time for both implementations is negligible because they are both very fast.

## 4 Proximity Subprotocols via Garbled Circuits

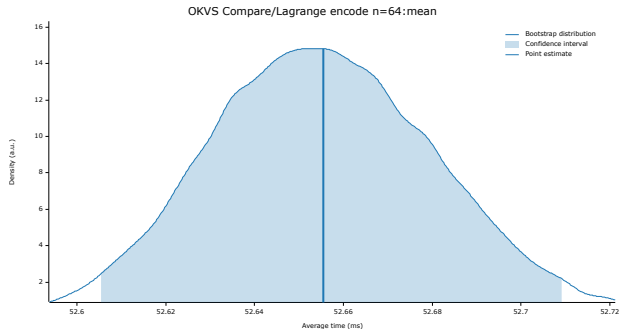
At the core of the fuzzy PSI protocol lies a generic *proximity subprotocol*, executed between parties for each candidate bin match. Its purpose is to determine whether two inputs $x$ and $y$ are "close enough" with respect to a public distance metric and threshold. To ensure security and modularity, these subprotocols are instantiated using *Yao's garbled circuits*.

In this setting, the circuit is hardcoded with one party's input and the distance threshold $\delta$, and evaluates whether the other party's input lies within that threshold. The result is revealed only to the initiating party, without disclosing the actual inputs. For different distance metrics, different garbling schemes are employed: arithmetic garbling [1] for $\ell_2$ (enabling efficient squaring), and boolean garbling [6] for $\ell_1$ and $\ell_\infty$.

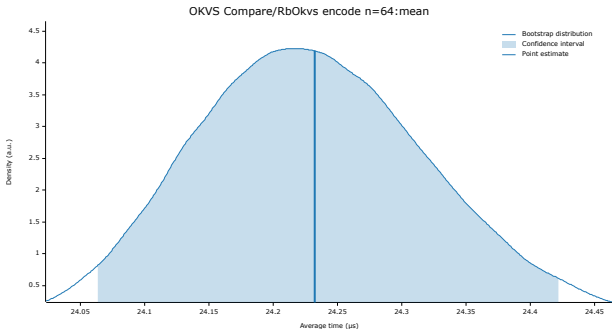
These subprotocols are designed to be pseudorandom and leak no information when inputs are far apart. They integrate seamlessly into the main protocol by being encoded into the OKVS structure, enabling secure and efficient pairwise proximity testing at scale.

Garbled circuits, while powerful, introduce significant computational and communication overhead. Each proximity test must be represented as a Boolean or arithmetic circuit, which is then garbled and evaluated using cryptographic protocols such as oblivious transfer. This process involves multiple rounds of encryption and the transmission of large garbled tables, making it much more resource-intensive than simple plaintext computation. In practice, this overhead can become a bottleneck, especially when many proximity tests must be performed in parallel as part of the fuzzy PSI protocol.

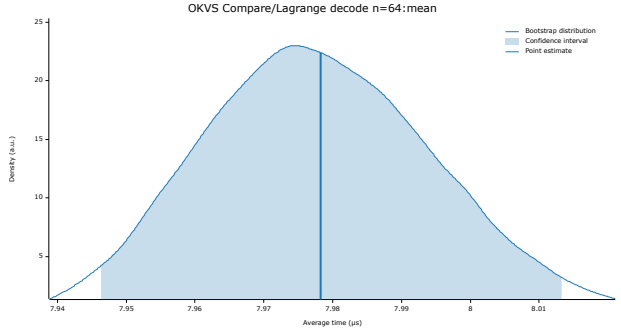
In this implementation, actual garbled circuits are not included. The main reason is the lack of a readily available, well-maintained Rust library for the specific proximity circuits required by the protocol. Implementing these circuits from scratch would have required substantial additional engineering effort, which was beyond the scope of this project. As a result, the current implementation uses a non-secure placeholder: instead of securely evaluating the proximity function, it simply shares



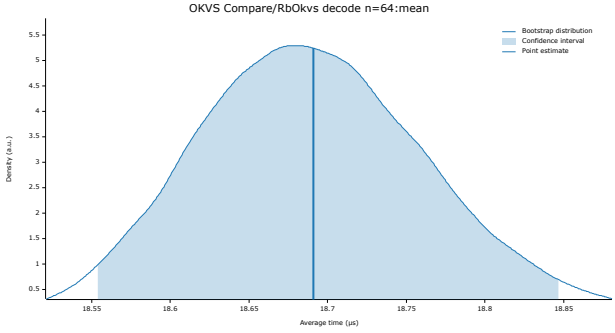

(a) Lagrange polynomial OKVS encoding with $n = 64$.



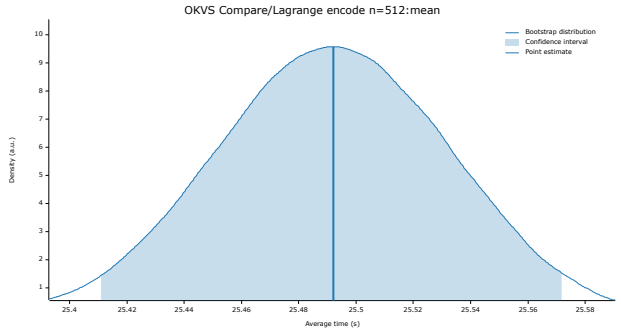

(b) RB-OKVS encoding with $n = 64$.



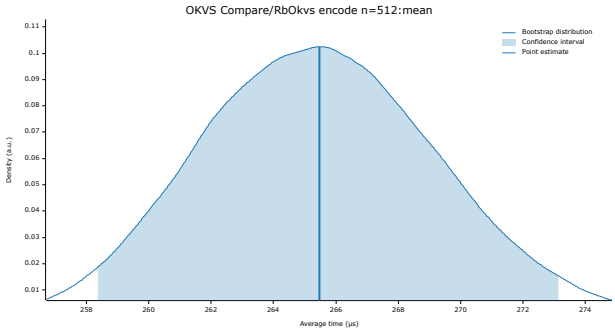

(c) Lagrange polynomial OKVS decoding with $n = 64$.



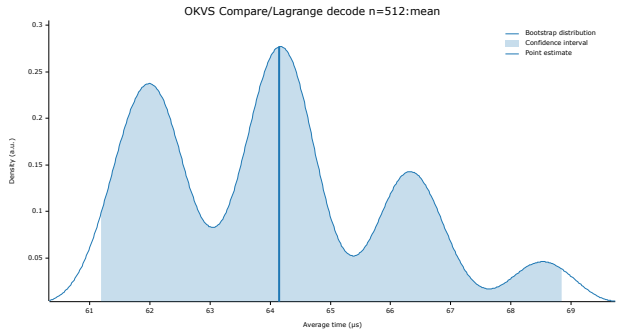

(d) RB-OKVS decoding with $n = 64$.



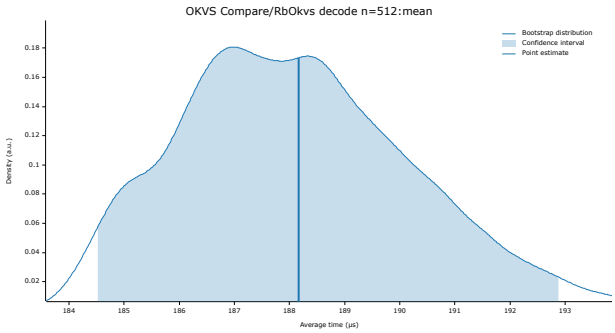

(e) Lagrange polynomial OKVS encoding with $n = 512$.




(f) RB-OKVS encoding with $n = 512$.




(g) Lagrange polynomial OKVS decoding with $n = 512$.




(h) RB-OKVS decoding with $n = 512$.

Figure 1: Encoding and decoding times for the Lagrange polynomial and RB-OKVS implementations with $n = 64$ and $n = 512$.

the actual points between the parties (as can be seen in the main protocol logic). This approach is definitely not secure and is only suitable for testing and benchmarking the rest of the protocol. In a real deployment, this component must be replaced with a proper secure computation protocol, such as Yao's garbled circuits or a similar cryptographic primitive.

# 5 Fuzzy PSI - Full Protocol Overview

The fuzzy PSI protocol by Richardson, Rosulek, and Xu [4] is designed as a modular framework that combines the three mentioned key cryptographic components: conditionally-overlapping hashing, oblivious key-value stores, and secure proximity subprotocols. These elements are orchestrated to enable two parties, each with private input sets, to identify approximately matching items without revealing any additional information. This section describes the high-level workflow and internal mechanics of the protocol.

## 5.1 Protocol Structure

The protocol operates under the following core principles:

- **Hashing and Binning:** Each party maps its inputs into a collection of bins using a conditionally-overlapping hash function. The binning ensures that any pair of inputs $(x, y)$ that are "close" under a public distance metric $d$ with threshold $\delta$ will end up in at least one common bin. Depending on the configuration, binning may be asymmetric or split across dimensions to balance communication costs.
- **Subprotocol Execution:** For each active bin, the parties run a secure *proximity subprotocol* to determine whether the items in that bin are within the similarity threshold. These subprotocols are instantiated using Yao's garbled circuits, which allow secure two-party computation of distance checks with minimal leakage.
- **OKVS Encoding:** To avoid revealing which bins are active, each party encodes its messages for each active bin into an *Oblivious Key-Value Store*. The OKVS hides the bin identifiers and ensures that only the relevant proximity test results can be accessed by the other party. This mechanism enables privacy-preserving message routing without interaction per bin.

## 5.2 Workflow Description

The protocol proceeds as follows (see Figure 2):

1. **Local Binning:** Each party computes the set of bins their items map to. Let $A$ be Alice's input set and $H_1$ her hash function, then she computes $H_1(A)$. Similarly, Bob computes $H_2(B)$ using his hash function.
2. **Subprotocol Preparation:** For each of Alice's active bins $\beta \in H_1(A)$, she computes a proximity subprotocol message $M_1[\beta]$ and retains internal state $\sigma[\beta]$. These messages encode the result of running `PROT1` on her input for that bin.
3. **OKVS Encoding (Alice):** Alice encodes the bin-to-message mapping $\{(\beta, M_1[\beta])\}$ into an OKVS, producing a polynomial or table $P$. She sends this to Bob.
4. **Subprotocol Response (Bob):** For each of his active bins $\beta' \in H_2(B)$, Bob decodes $M_1[\beta']$ from the received OKVS and runs `PROT2` with his input to obtain $M_2[\beta']$.
5. **OKVS Encoding (Bob):** Bob encodes his responses into a second OKVS $Q$ and sends it back to Alice.

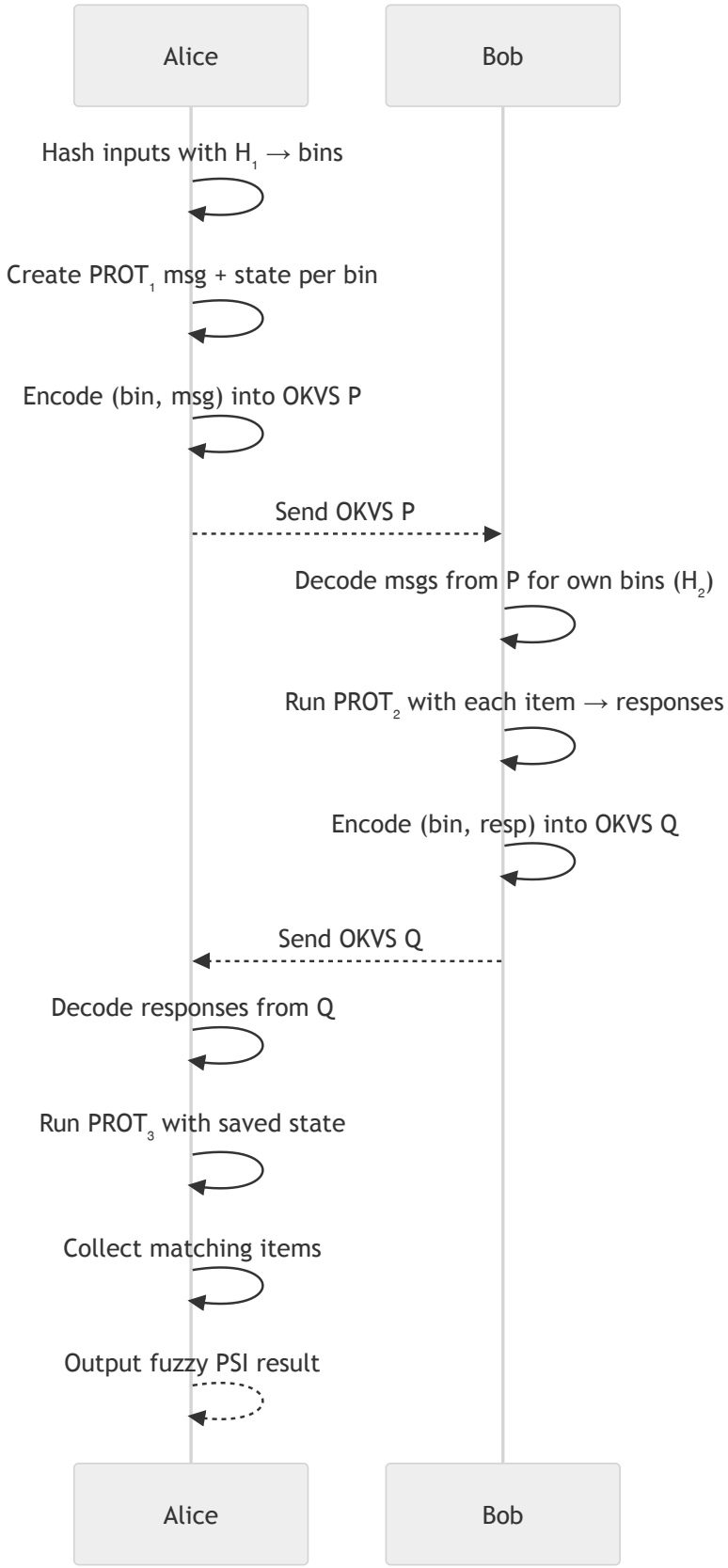



Figure 2: Flow of the fuzzy PSI protocol.

6. **Subprotocol Finalization (Alice):** For each of her bins $\beta \in H_1(A)$, Alice decodes $M_2[\beta]$ from $Q$ and uses `PROT3` with her stored $\sigma[\beta]$ to compute the result of the proximity test. If a match is detected, she records the result and associates it with the corresponding bin.

### 5.3 Security and Privacy Guarantees

Throughout the protocol, privacy is maintained by three layers:

- **Hashing ensures locality, not exposure:** Items are mapped deterministically to bins, but the actual bin identifiers are concealed within the OKVS.
- **OKVS obfuscates structure:** The use of oblivious encodings hides the set of active bins and prevents inference of item presence or density patterns.
- **Subprotocols protect values:** Yao's garbled circuits ensure that only proximity (not actual distances or values) is learned, and only by the initiating party.

### 5.4 Generalization and Efficiency

The modular nature of the protocol allows it to support a wide range of distance metrics, including $\ell_1$, $\ell_2$, and $\ell_\infty$, by adjusting the hash functions and the instantiation of the proximity subprotocol. The authors demonstrate that for moderate or large values of $\delta$, their protocol achieves best-in-class communication efficiency compared to other fuzzy PSI constructions.

In summary, the protocol achieves fuzzy private set intersection through secure, distance-aware hashing, compact and oblivious message routing via OKVS, and precise approximate matching using optimized proximity subprotocols.

## 6 Reflection

This project was both fun and interesting to work on. Collaborating with Florias was a great experience, as we figured out together how the protocol actually works and how its components fit together. The initial phase of understanding the protocol was intellectually rewarding and provided valuable insights into modern cryptographic constructions for fuzzy private set intersection.

However, during the implementation phase, it became clear that the protocol would be challenging to realize in practice. The first major hurdle was finding a suitable implementation of the Oblivious Key-Value Store. The provided RB-OKVS code was very old and unmaintained, so I initially attempted my own implementation. The Lagrange-based OKVS worked, but it was extremely slow and impractical for larger set sizes. I then switched back to the RB-OKVS, investing significant effort to update and fix the legacy code so it would work with modern Rust and the needs of this project.

Once the OKVS component was functional, I encountered the next major challenge: the protocol's subprotocols rely on Yao's garbled circuits for secure proximity testing. Unfortunately, there was no ready-to-use, well-maintained implementation of these circuits available in Rust. After some trial and error, we decided to proceed with an insecure version of the protocol that simply shares the actual points between parties. While this allowed us to test and benchmark the OKVS components, it means that the overall protocol cannot be securely benchmarked or deployed as intended.

In conclusion, this was a very fun project to work on, but the nature of the required subprotocols makes a full, practical implementation difficult. The performance of the protocol, especially with the current cryptographic building blocks, is generally poor and not yet fit for practical usage. Nonetheless, the project provided valuable experience in cryptographic engineering, protocol design, and the realities of implementing advanced privacy-preserving technologies.

## References


[1] Marshall Ball, Tal Malkin, and Mike Rosulek. "Garbling Gadgets for Boolean and Arithmetic Circuits". In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. CCS '16. New York, NY, USA: Association for Computing Machinery, Oct. 24, 2016, pp. 565–577. ISBN: 978-1-4503-4139-4. DOI: 10.1145/2976749.2978410.

[2] Chongwon Cho, Dana Dachman-Soled, and Stanisław Jarecki. "Efficient Concurrent Covert Computation of String Equality and Set Intersection". In: *Topics in Cryptology - CT-RSA 2016*. Ed. by Kazue Sako. Cham: Springer International Publishing, 2016, pp. 164–179. ISBN: 978-3-319-29485-8. DOI: 10.1007/978-3-319-29485-8_10.

[3] Ying Gao et al. "Efficient Fuzzy Private Set Intersection from Fuzzy Mapping". In: *Advances in Cryptology – ASIACRYPT 2024*. Ed. by Kai-Min Chung and Yu Sasaki. Singapore: Springer Nature, 2025, pp. 36–68. ISBN: 978-981-96-0938-3. DOI: 10.1007/978-981-96-0938-3_2.

[4] David Richardson, Mike Rosulek, and Jiayu Xu. *Fuzzy PSI via Oblivious Protocol Routing*. 2024. URL: https://eprint.iacr.org/2024/1642 (visited on 12/18/2024). Pre-published.

[5] Aron van Baarsen and Sihang Pu. "Fuzzy Private Set Intersection with Large Hyperballs". In: *Advances in Cryptology – EUROCRYPT 2024*. Ed. by Marc Joye and Gregor Leander. Cham: Springer Nature Switzerland, 2024, pp. 340–369. ISBN: 978-3-031-58740-5. DOI: 10.1007/978-3-031-58740-5_12.

[6] Samee Zahur, Mike Rosulek, and David Evans. "Two Halves Make a Whole". In: *Advances in Cryptology - EUROCRYPT 2015*. Ed. by Elisabeth Oswald and Marc Fischlin. Berlin, Heidelberg: Springer, 2015, pp. 220–250. ISBN: 978-3-662-46803-6. DOI: 10.1007/978-3-662-46803-6_8.

## Declaration of Authorship

I (Christoph Bühler) hereby declare,

- that I have written this report independently,
- that I have written the report using only the aids specified in the index;
- that all parts of the report produced with the help of aids have been declared;
- that I have handled both input and output responsibly when using AI. I confirm that I have therefore only read in public data or data released with consent and that I have checked, declared and comprehensibly referenced all results and/or other forms of AI assistance in the required form and that I am aware that I am responsible if incorrect content, violations of data protection law, copyright law or scientific misconduct (e.g. plagiarism) have also occurred unintentionally;
- that I have mentioned all sources used and cited them correctly according to established academic citation rules;
- that I have acquired all immaterial rights to any materials I may have used, such as images or graphics, or that these materials were created by me;
- that the topic, the report or parts of it have not already been the object of any work or examination of another course, unless this has been expressly agreed with the faculty member in advance and is stated as such in the report;
- that I am aware of the legal provisions regarding the publication and dissemination of parts or the entire report and that I comply with them accordingly;
- that I am aware that my report can be electronically checked for plagiarism and for third-party authorship of human or technical origin and that I hereby grant the University of St. Gallen the copyright according to the Examination Regulations as far as it is necessary for the administrative actions;
- that I am aware that the University will prosecute a violation of this Declaration of Authorship and that disciplinary as well as criminal consequences may result, which may lead to expulsion from the University or to the withdrawal of my title.

By submitting this report, I confirm through my conclusive action that I am submitting the Declaration of Authorship, that I have read and understood it, and that it is true.

## Directory of writing aids

| **Aid** | **Usage / Application** | **Affected Areas** |
|---|---|---|
| LanguageTool | Spellcheck | Whole document |
| Google Scholar | Literature Research | References |
| ChatGPT | Spellcheck and Correction | Whole document |

Table 1: Writing Aids (Art. 57 AB)