FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE D COIMBRA

Secure Multiparty Computation

Segurança e Privacidade • 2025/2026

Relatório

Elementos do Grupo

Daniel Pereira • 2021237092 • uc2021237092@student.uc.pt

Vasco Fernandes • 2025186614 • uc2025186614@student.uc.pt

Índice

1. Introdução

Nos dias de hoje, a partilha de informação entre diferentes entidades é essencial para diversas aplicações, como sistemas financeiros, instituições de saúde ou empresas que pretendem apenas identificar clientes em comum. No entanto, esta troca de dados pessoais levanta sérias preocupações de privacidade e proteção de dados, uma vez que a divulgação direta de informação pode violar políticas de confidencialidade e regulamentos, como o RGPD (Regulamento Geral sobre a Proteção de Dados).

1.1. Secure Multiparty Computation

Para enfrentar esse desafio, surgiram mecanismos de *Secure Multiparty Computation*, uma área da criptografia que permite que várias partes realizem de forma cooperativa o cálculo de uma função sem revelar os seus dados individuais. Em vez de partilhar os dados originais, cada participante realiza operações sobre versões cifradas ou mascaradas de informação, garantindo que apenas o resultado final é conhecido [1].

O SMC baseia-se em princípios fundamentais de confidencialidade, correção e independência da informação:

- Confidencialidade: nenhuma das partes deve aprender informação adicional além do resultado acordado.
- Correção: o resultado da computação deve ser exato, mesmo que as partes não confiem umas nas outras.
- Independência da entrada: cada participante deve introduzir os seus dados sem ser influenciado pelas entradas dos outros.

1.2. Private Set Intersection

Entre os diversos protocolos que implementam estes princípios, destaca-se o *Private Set Intersection* (PSI), que tem como objetivo determinar quais elementos são comuns entre os conjuntos de dados de duas partes, sem revelar quaisquer outros elementos não coincidentes. Este tipo de protocolo é amplamente aplicável em contextos onde há necessidade de cooperação entre entidades — por exemplo, hospitais que procuram identificar pacientes comuns sem violar a privacidade dos registos clínicos.

Neste projeto foram estudadas várias abordagens de PSI, integradas no contexto de *Secure Multiparty Computation,* de modo a compreender o seu funcionamento, segurança e desempenho. Foram analisadas quatro variantes principais: o Naïve PSI, o Server-Aided PSI, o Diffie-Hellman PSI e o OT-Based PSI. Cada uma destas variantes representa um equilíbrio diferente entre eficiência computacional e grau de privacidade.

1.3. Ferramentas Utilizadas

A análise experimental foi conduzida utilizando ferramentas fornecidas (demo.exe e psi.exe) complementadas por capturas de tráfego em rede com o *Wireshark*, permitindo observar e comparar o comportamento de cada protocolo em cenários de diferentes

tamanhos de conjuntos de dados. Todo o processo foi realizado numa máquina virtual *Lubuntu 20.04.4 Focal Fossa*, recorrendo ao PSI disponível em https://github.com/bluetrickpt/PSI.

A compilação e execução dos binários foi realizada em ambiente Linux, com as seguintes dependências:

- g++ — compilador C++
- make — sistema de build
- libgmp-dev — operações aritméticas de grande precisão
- libglib2.0-dev e libssl-dev — utilizada para operações criptográficas

O objetivo final é avaliar a forma como o PSI concretiza os princípios de *Secure Multiparty Computation* na prática, garantindo privacidade sem comprometer a utilidade da informação partilhada.

2. Definição do Problema e dos Conjuntos de Dados

Nos tempos atuais, as empresas dependem fortemente da análise de dados para compreender os seus clientes, melhorar serviços e desenvolver estratégias de *marketing* eficazes. Em particular, a colaboração entre diferentes organizações pode permitir campanhas mais direcionadas e relevantes, através da identificação de utilizadores comuns entre as bases de dados das empresas.

No entanto, a partilha direta de dados pessoais entre empresas levanta sérias preocupações de privacidade e riscos legais. O Regulamento Geral sobre a Proteção de Dados (RGPD), em vigor na União Europeia desde 2018, impõe restrições rigorosas ao tratamento e troca de dados pessoais, exigindo que qualquer partilha seja [2]:

- "Justificada por uma finalidade legítima,"
- "autorizada pelo titular dos dados, e"
- "realizada com garantias técnicas adequadas de confidencialidade e segurança."

Assim, torna-se necessário encontrar soluções que permitam a cooperação entre entidades sem comprometer a privacidade dos utilizadores

2.1. Cenário Prático

Neste projeto, considera-se um cenário realista do setor comercial, no qual duas empresas de serviços distintos pretendem colaborar para identificar clientes comuns, sem comprometer a privacidade dos dados pessoais:

- **Empresa A**: uma operadora de telecomunicações que fornece serviços de internet e televisão

- **Empresa B**: uma plataforma de streaming de vídeo e música online, que disponibiliza conteúdos por meio de subscrições mensais.

Ambas as empresas planeiam desenvolver uma campanha conjunta de fidelização, oferecendo descontos ou pacotes promocionais a clientes que possuam contratos ativos com ambas as entidades.

Cada empresa mantém uma base de dados própria, estruturada com os seguintes atributos:

Campo	Descrição
uid	Identificador único em cada dataset
full_name	Nome completo
email	Endereço de email
phone	Número de telemóvel
dob	Data de nascimento
country	País de residência

Tabela 1- Estrutura dos Datasets

Exemplo: UID00000246,Daniel Williams,daniel.williamscommon@inbox.com,+38 865 590 319,1965-09-09,Italy

Estes datasets foram gerados a partir de um script em Python, o objetivo deste script é gerar conjuntos de dados artificiais (datasets) para testar o protocolo PSI – Private Set Intersection, simulando dois participantes ("Party A" e "Party B") com um certo número de registos e uma percentagem de elementos em comum.

Com este método é possível:

- Controlar o nível de interseção entre os conjuntos;
- Estudar a privacidade e anonimato dos dados;
- Demonstrar a eficiência e escalabilidade do protocolo PSI.

Foi escolhido este tipo de dataset por representar um cenário realista de partilha de dados pessoas entre organizações, contendo atributos comuns em bases de clientes. Estes datasets encontram-se armazenados em ficheiros .csv, que contêm listas de clientes recolhidas de forma independente, mas com alguma sobreposição.

O objetivo é determinar quais os clientes comuns às duas empresas, de modo a permitir a criação de ofertas conjuntas sem expor os dados restantes.

2.2. Datasets

Após a definição da estrutura base do dataset, foram gerados cinco versões dos ficheiros de dados, variando apenas no número de registos incluídos.

Esta abordagem permite avaliar o impacto da dimensão dos dados no desempenho e na eficiência dos diferentes protocolos PSI.

Os tamanhos foram escolhidos de forma progressiva e proporcional, de modo a observar a escalabilidade do sistema em diferentes ordens de grandeza — desde um conjunto reduzido até um volume elevado de dados. Assim, é possível medir como o tempo de execução e o volume de comunicação evoluem com o crescimento do dataset.

Os tamanhos específicos utilizados foram:

- Versão 1: 5 000 registos
- Versão 2: 10 000 registos
- Versão 3: 20 000 registos
- Versão 4: 40 000 registos
- Versão 5: 80 000 registos

Esta escolha equilibra realismo e viabilidade experimental — valores suficientemente grandes para representar cenários reais de interseção de dados entre entidades, mas ainda exequíveis em ambiente de teste e dentro de tempos de execução controlados.

3. Análise teórica dos Protocolos PSI

Antes de passarmos à análise experimental é importante perceber o funcionamento conceptual de cada protocolo, as suas vantagens, limitações e nível de segurança [3].

3.1. Naïve PSI (Protocolo 0)

O Naïve PSI é a versão mais simples do protocolo, baseada apenas em funções de hash. Cada parte aplica uma função de hash a todos os seus elementos e envia as versões cifradas à outra parte. A interseção é então calculada comparando os valores de hash coincidentes.

Este método é simples e eficiente em termos de implementação, porém não oferece segurança adequada. Quando os dados têm baixa entropia, um adversário pode adivinhar os valores originais e inverter o hash, obtendo informações sobre os elementos do conjunto.

Por esse motivo, o Naïve PSI é adequado apenas para cenários de teste e demonstração, e não para aplicações reais que envolvam casos sensíveis.

3.2. Server-Aided PSI (Protocolo 1)

O Server-Aided PSI introduz uma terceira entidade designada TTP (Trusted Third Party), que auxilia o cliente e o servidor na computação da interseção.

O objetivo deste modelo é reduzir o custo computacional para as duas partes principais, no entanto, o sucesso e segurança deste protocolo dependem fortemente da confiança nesta entidade. Assume-se que o TTP é honesto e não colabora com nenhuma das partes. Por este motivo, este tipo de PSI é teoricamente interessante, mas pouco prático em sistemas reais distribuídos, dado o risco inerente de quebra de confiança.

3.3. Diffie-Hellman PSI (Protocolo 2)

O Diffie-Hellman PSI utiliza as propriedades matemáticas de exponenciação modular, explorando o facto de $(x^y)^z = (x^z)^y$.

Neste protocolo, cada parte eleva os elementos do seu conjunto a um expoente secreto, troca os resultados e aplica novamente a exponenciação com o seu próprio expoente. Assim, ambas conseguem identificar os elementos comuns sem revelar os restantes.

Embora a implementação seja simples, o método assume um elevado nível de confiança entre as partes e não protege totalmente a privacidade quando os dados têm baixa entropia ou quando há um comportamento malicioso.

3.4. OT-Based PSI (Protocolo 3)

O OT-Based PSI (baseado em Oblivious Transfer) representa uma das implementações mais seguras e modernas do PSI. O conceito de Oblivious Transfer permite que uma das partes (o cliente) obtenha certas informações da outra parte (o servidor) sem que o servidor saiba qual informação foi recebida, e sem que o cliente aprenda mais do que o permitido. Este protocolo combina técnicas de Oblivious Transfer com hashing e compressão criptográfica, garantindo que apenas os elementos coincidentes são revelados.

É o mais complexo de implementar, mas também o mais seguro. Por essa razão, o OT-Based PSI é amplamente utilizado em aplicações industriais onde a privacidade é crítica.

4. Recolha de Dados

Inicialmente, todos os protocolos PSI foram testados manualmente, de modo a compreender o funcionamento interno de cada abordagem e o papel das diferentes partes envolvidas. Cada protocolo foi executado em instâncias distintas do terminal, representando as entidades participantes no processo de interseção.

Todos os testes foram executados em localhost, removendo a existência de latência real de rede e eliminado todos os fatores externos, o que pode ter contribuído para tempos inferiores aos que seriam observados num ambiente distribuído.

Nos protocolos de duas partes —Naïve PSI, Diffie-Hellman PSI e OT-Based PSI — foram utilizadas duas instâncias de terminal na mesma máquina virtual. Em cada execução, uma instância assumia o papel de servidor (com o parâmetro -r 0), responsável escutar a ligação na porta 7766, enquanto a outra instância atuava como cliente (com o parâmetro -r 1), iniciando a ligação e enviando o conjunto de dados correspondente. Ambas as partes executavam o mesmo binário demo.exe, diferenciando-se apenas pelo papel atribuído e pelo ficheiro de entrada.

No caso do Server-Aided PSI, foi necessária uma terceira instância de terminal, correspondente à Trusted Third Party. Nesta configuração, o servidor TTP era iniciado

primeiro (com -r 0 e – p 1), seguido pelas duas partes participantes (-r 1). Cada cliente enviava os seus dados cifrados à TTP, que realizava a operação de interseção e devolvia o resultado.

4.1. Interseções

Em todos os protocolos avaliados o número de interseções observadas manteve-se constante para cada dimensão dos datasets, tal como era esperado. Esta consistência demonstra que o processo foi corretamente estruturado, assegurando uma taxa de interseção de 20% entre as listas da Party A e da Party B.

Tamanho do dataset	Interseções
5000	1000
10000	2000
20000	4000
40000	8000
80000	16000

Tabela 2 - Interseções por tamanho de dataset

4.2. Tempos de execução

Para otimizar o processo experimental de recolha de tempos de execução, foi desenvolvido um script bash para cada protocolo. Este script executa automaticamente o respetivo protocolo para todas as versões dos datasets (5 000, 10 000, 20 000, 40 000 e 80 000 registos). Os scripts utilizados podem ser consultados no Anexo A.

Para cada protocolo PSI e para cada versão do dataset foram realizados três testes, com o objetivo de reduzir impacto de variações momentâneas no sistema.

4.2.1. Naïve PSI

Pela análise da Tabela 3, percebe-se que o tempo de execução cresce de quase forma linear com o aumento do tamanho do dataset, evidenciando o comportamento previsível e estável do protocolo. Mesmo para o maior dataset, o tempo médio não ultrapassa 0,6 segundos, o que confirma a elevada eficiência do método. Este desempenho deve-se à simplicidade do algoritmo, que apenas aplica funções de hashing aos elementos de cada conjunto, sem recorrer a funções criptográficas complexas.

Protocolo	Dataset	Execução 1 (s)	Execução 2 (s)	Execução 3 (s)	Média (s)
0 (Naïve)	5000	0,056	0,065	0,056	0,0590
0 (Naïve)	10000	0,093	0,089	0,088	0,0900
0 (Naïve)	20000	0,18	0,18	0,088	0,1493
0 (Naïve)	40000	0,286	0,298	0,305	0,2963
0 (Naïve)	80000	0,582	0,569	0,554	0,5683

Tabela 3 - Análise temporal do Naïve

4.2.2. Server Aided PSI

Os resultados obtidos para o protocolo Server Aided PSI encontram-se na Tabela 4, apresentando uma evolução aparentemente linear com o aumento do tamanho dos

datasets. O tempo médio de execução variou de 0,086 para 5 000 registos até 0,843 para 80 000 registos, confirmando um crescimento proporcional ao tamanho dos dados.

Protocolo	Dataset	Execução 1 (s)	Execução 2 (s)	Execução 3 (s)	Média (s)
1 (Server-Aided)	5000	0,088	0,086	0,085	0,0863
1 (Server-Aided)	10000	0,142	0,122	0,112	0,1253
1 (Server-Aided)	20000	0,237	0,229	0,243	0,2363
1 (Server-Aided)	40000	0,429	0,454	0,444	0,4423
1 (Server-Aided)	80000	0,836	0,848	0,845	0,8430

Tabela 4 - Análise temporal do Server Aided

4.2.3. Diffie-Hellman PSI

Os resultados para o protocolo Diffie-Hellman PSI encontram-se resumidos na Tabela 5, evidenciando um aumento significativo face aos dois protocolos vistos anteriormente. O tempo aumentou progressivamente de 5 segundos para 5 000 registos até cerca de 100 segundos. Esta diferença é devida ao elevado custo computacional das operações criptográficas.

Protocolo	Dataset	Execução 1 (s)	Execução 2 (s)	Execução 3 (s)	Média (s)
2 (Diffie-Hellman)	5000	4,485	6,323	4,464	5,0907
2 (Diffie-Hellman)	10000	8,896	12,982	9,09	10,3227
2 (Diffie-Hellman)	20000	18,173	25,496	18,613	20,7607
2 (Diffie-Hellman)	40000	38,187	53,203	45,754	45,7147
2 (Diffie-Hellman)	80000	102,35	100,914	98,702	100,6553

Tabela 5 - Análise temporal do Diffie-Hellman

4.2.4. OT-Based PSI

Através da análise da Tabela 6, deparamo-nos com uma grande descida de tempos médios comparando com o protocolo Diffie-Hellman PSI, resultado inesperado que é analisado mais à frente.

Protocolo	Dataset	Execução 1 (s)	Execução 2 (s)	Execução 3 (s)	Média (s)
3 (OT-Based)	5000	0,328	0,346	0,318	0,3307
3 (OT-Based)	10000	0,38	0,379	0,365	0,3747
3 (OT-Based)	20000	0,433	0,429	0,415	0,4257
3 (OT-Based)	40000	0,531	0,527	0,51	0,5227
3 (OT-Based)	80000	0,71	0,684	0,701	0,6983

Tabela 6 - Análise temporal do OT-Based

4.3. Recolha de dados de Tráfego de Rede com Wireshark

Para complementar a análise de desempenho temporal dos protocolos PSI, procedeu-se à captura e análise do tráfego de rede gerado utilizando a ferramenta Wireshark. O objetivo desta etapa foi quantificar o volume de dados trocados entre as partes e avaliar a eficiência da comunicação associada a cada abordagem.

As capturas foram realizadas na interface loopback (127.0.0.1), dado que todas as execuções ocorreram localmente na mesma máquina virtual. Antes de iniciar cada

execução, foi aplicado o filtro tcp.port == 7766, correspondente à porta de comunicação utilizada pelo programa demo.exe, garantindo que apenas o tráfego relacionado com os protocolos PSI fosse capturado.

Após feitas as médias foram alcançados os resultados presentes na Tabela 7.

Protocolo	Dataset	Total (B)	Pacotes
0 (Naïve)	5000	91838	27
0 (Naïve)	10000	182102	31
0 (Naïve)	20000	362696	40
0 (Naïve)	40000	723620	54
0 (Naïve)	80000	1606986	105
1 (Server-Aided)	5000	164012	41
1 (Server-Aided)	10000	326120	54
1 (Server-Aided)	20000	650138	77
1 (Server-Aided)	40000	1297646	115
1 (Server-Aided)	80000	2227064	144
2 (Diffie-Hellman)	5000	566070	49
2 (Diffie-Hellman)	10000	1097798	75
2 (Diffie-Hellman)	20000	2160624	118
2 (Diffie-Hellman)	40000	4317298	177
2 (Diffie-Hellman)	80000	7043896	250
3 (OT-Based)	5000	551348	61
3 (OT-Based)	10000	1081280	87
3 (OT-Based)	20000	2106396	109
3 (OT-Based)	40000	4191112	179
3 (OT-Based)	80000	7544100	256

A análise quantitativa das capturas com o Wireshark mostrou que o volume de dados trocados aumenta conforme o aumento do tamanho dos datasets em todos os protocolos PSI. O Naïve PSI foi o mais eficiente em termos de comunicação, apresentando o menor número de pacotes e bytes trocados enquanto o OT-Based PSI atingiu 256 pacotes e 7,5 MB trocados. No tópico seguinte foi realizada uma análise mais detalhada dos resultados obtidos, permitindo avaliar com maior precisão o comportamento dos diferentes protocolos.

Tabela 7 - Resultados da Análise com o Wireshark

5. Resultados

5.1. Tamanho do dataset vs. Tempo de execução

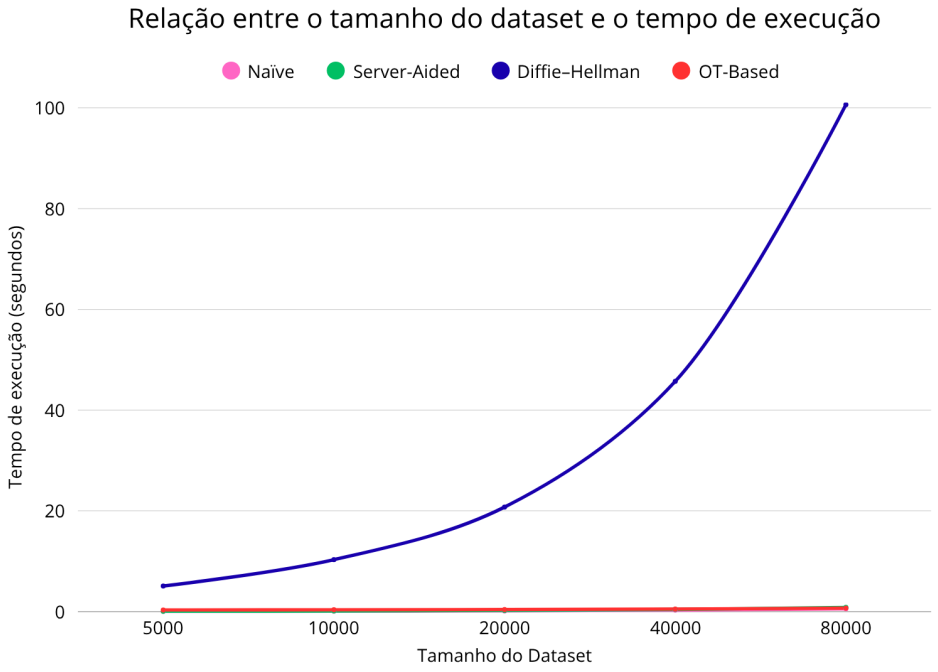

Gráfico 1.A – Gráfico que compara o tamanho do dataset com o tempo de execução

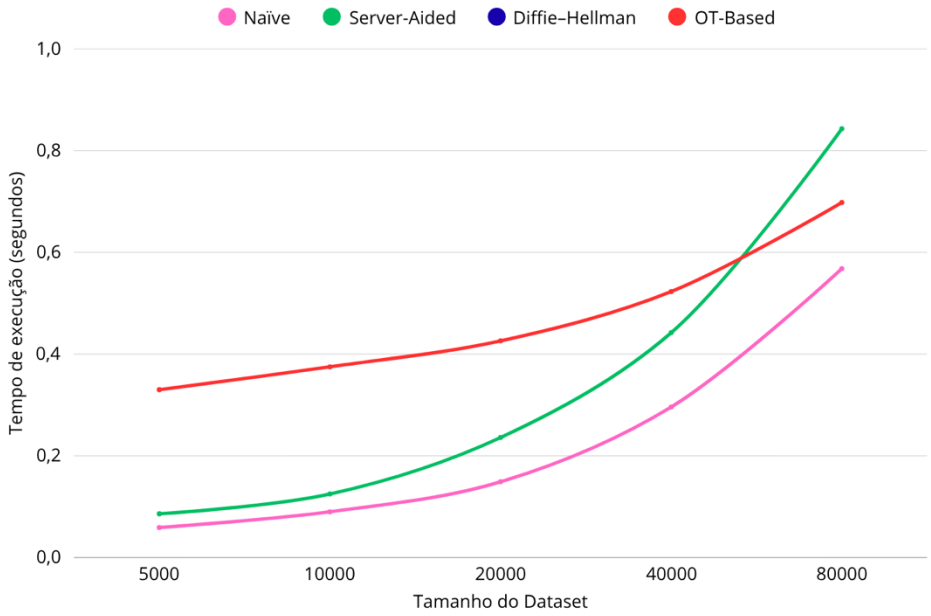

Gráfico 1.B – Aproximação para melhor observação dos protocolos Naïve, Server-Aided e OT-Based

O Gráfico 1 apresenta a relação entre o tamanho dos datasets e o tempo médio de execução dos quatro protocolos PSI avaliados. A parte A do gráfico mostra a visão geral, incluindo as curvas dos quatro protocolos, enquanto a parte B apresenta uma ampliação

que permite observar com melhor detalhe o comportamento das curvas dos protocolos mais rápidos (tempos < 1 segundo).

Da análise dos gráficos, apercebemo-nos de uma tendência de crescimento acentuado nos protocolos Naïve e Server-Aided PSI. Esta aproximação com a linearidade indica que o custo temporal aumenta de forma proporcional à dimensão dos dados analisados, evidenciando uma boa escalabilidade.

O Naïve PSI é o mais rápido, com tempos médios entre os 0,06 e os 0,57 segundos, resultado da sua simplicidade.

O OT-Based PSI exibe um comportamento interessante do ponto de vista teórico, pois apesar da sua complexidade, apresenta tempos semelhantes aos dos protocolos mais simples (entre 0,33 e 0,69 s).

Já o Diffie-Hellman PSI destaca-se dos restantes devido ao crescimento quase exponencial do tempo de execução. Os valores variam entre 5,09 segundos até 1 minuto e 40 segundos.

5.2. Tamanho do dataset vs. Dados trocados

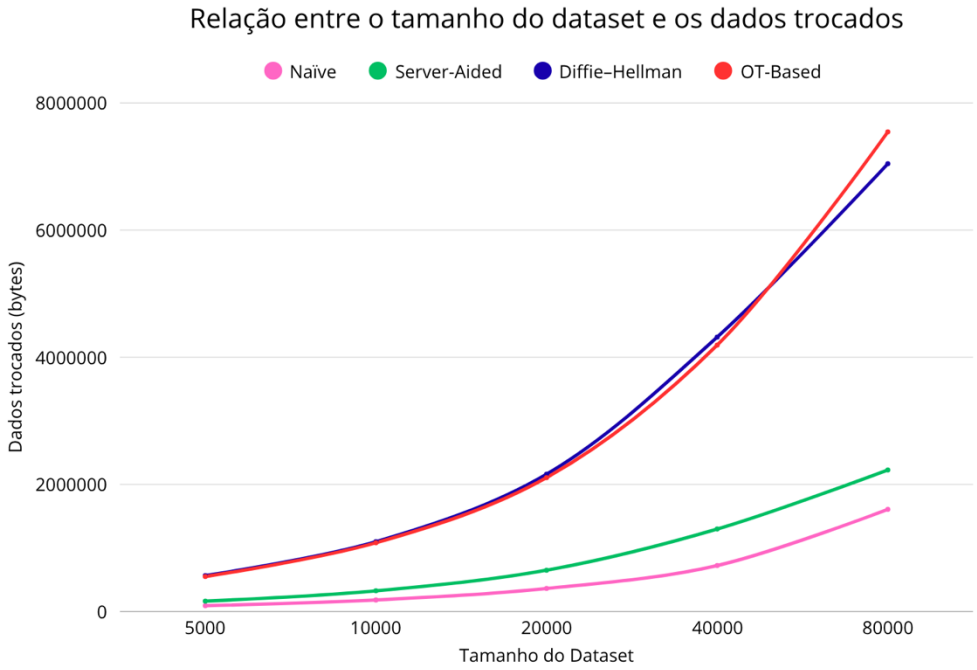

Gráfico 2 - Gráfico que compara o tamanho do dataset com o número de bytes trocados

O Gráfico 2 ilustra a relação entre o tamanho dos datasets e o volume total de dados trocados entre as partes em cada protocolo PSI, obtidos a partir das capturas de tráfego realizadas com recurso ao Wireshark.

Os protocolos Naïve e Server-Aided PSI exibem curvas quase iguais que crescem de maneira aproximadamente linear em relação ao tamanho do dataset. A diferença entre os dois é mínima e deve-se sobretudo à presenta da TTP.

Os protocolos Diffie-Hellman e OT-Based PSI também apresentam curvas bastante semelhantes entre si, caracterizadas por um crescimento bastante acentuado.

6. Discussão dos resultados

A partir dos resultados experimentais recolhidos e apresentados nas secções anteriores, é possível compreender as razões técnicas que explicam as diferenças de desempenho entre os quatro protocolos PSI avaliados.

Essas diferenças resultam fundamentalmente de três circunstâncias principais:

- 1. Complexidade computacional das operações criptográficas
- 2. Quantidade de mensagens trocadas entre as partes
- 3. Arquitetura e número de entidades envolvidas na comunicação

6.1. Naïve PSI

O Naïve apresentou o melhor desempenho em termos de tempo e tráfego de rede, consequências diretas da sua simplicidade estrutural.

Neste protocolo cada uma das duas partes aplica uma função de hash a todos os seus elementos e envia os resultados à outra parte. A interseção é obtida comparando os hashes coincidentes, sem operações criptográficas pesadas.

Por outro lado, esta simplicidade implica baixa segurança, pois um adversário pode tentar inverter os hashes por força bruta ou inferir informação através de ataques de dicionário.

Assim, o comportamento quase linear do tempo e do tráfego é previsível, dado que o custo cresce de forma proporcional ao número de elementos processados, mas à custa do falta de privacidade.

6.2. Server-Aided PSI

O Server-Aided PSI demonstrou um desempenho muito próximo ao do Naïve, tanto em tempo de execução como em volume de dados trocados, com apenas um ligeiro acréscimo.

Isso deve-se ao facto deste modelo introduzir uma terceira identidade — a Trusted Third Party — responsável por realizar a interseção. Embora a presença desta entidade adicione duas comunicações adicionais, o processamento efetuado pela TTP é relativamente simples e não aumenta significativamente o tempo total nem a quantidade de dados trocados.

Contudo, a confiança depositada na TTP constitui o principal ponto fraco do protocolo: se a TTP for comprometida, toda a privacidade é perdida.

6.3. Diffie-Hellman PSI

O Diffie-Hellman PSI foi o protocolo que apresentou o pior desempenho temporal e um volume de dados elevado. Esta diferença é justificada pela natureza deste algoritmo: o protocolo baseia-se em operações de exponenciação modular e em múltiplas rondas de comunicação para troca de resultados. Essas operações são computacionalmente dispendiosas, especialmente quando aplicadas a dezenas de milhares de elementos.

Apesar deste custo elevado, o Diffie-Hellman já oferece uma melhor segurança, o único problema é a necessidade de um elevado nível de confiança entre as partes.

6.4. OT-Based PSI

O OT-Based PSI apresentou um compromisso ideal entre desempenho temporal e segurança. Embora seja teoricamente mais complexo do que o Diffie-Hellman, a sua implementação prática utilizou técnicas de OT Extension, que permitem reduzir o número de operações criptográficas pesadas (como exponenciações ou encriptações assimétricas). Estas extensões permitem gerar múltiplas transferências OT a partir de um pequeno número inicial de OTs baseadas em criptografia de chave pública, substituindo as restantes por operações simétricas muito mais leves [5].

Consequentemente, o tempo de execução é significativamente inferior ao esperado no início do trabalho prático.

A curva de tráfego é quase idêntica à do Diffie-Hellman porque ambos trocam quantidades comparáveis de informação criptográfica, mas o OT-Based processa esses dados de forma bem mais eficiente.

7. Conclusão

O presente trabalho prático permitiu compreender na prática como os princípios de *Secure Multiparty Computation* podem ser aplicados para resolver problemas reais de partilha de informação com preservação de privacidade, através da avaliação comparativa de quatro variantes do protocolo *Private Set Intersection.*

A análise experimental demonstrou que os diferentes protocolos representam compromissos distintos entre desempenho e segurança.

O Naïve PSI revelou-se mais rápido e leve, mas com segurança praticamente inexistente, sendo apenas adequado para testes.

O Server-Aided PSI manteve tempos e volumes de tráfego semelhantes, beneficiando da simplicidade de execução, mas introduzindo uma dependência da TTP, o que limita a sua execução em ambientes reais.

Já o Diffie-Hellman PSI apresentou o pior desempenho computacional, consequência direta do elevado custo das operações de exponenciação modular, embora ofereça uma segurança superior aos anteriores.

Por fim, o OT-Based PSI destacou-se como o protocolo mais equilibrado e eficiente, alcançando tempos de execução surpreendentemente baixos. Esse desempenho resulta do uso de técnicas de OT Extension, que reduzem drasticamente o número de operações.

Os resultados obtidos confirmam que a escolha do protocolo PSI ideal depende do contexto e das exigências de privacidade.

Concluindo, o protocolo OT-Based PSI revelou-se a opção mais equilibrada, combinando segurança forte e bom desempenho.

Referências

[1] *GEEKSFORGEEKS*. *What is Secure Multiparty Computation?* Disponível em: https://www.geeksforgeeks.org/blogs/what-is-secure-multiparty-computation/. Acesso em: 9 nov. 2025.

[2] Regulamento Geral sobre a Proteção de Dados. Disponível em: https://eur-lex.europa.eu/PT/legal-content/summary/general-data-protection-regulation-gdpr.html. Acesso em: 10 nov. 2025

[3] *Secure Multiparty Computation & Privacy* , Slides, Segurança e Privacidade. Versão 2025-2026

[4] PSI, Readme. https://github.com/rscmendes/PSI/blob/master/README.md. Acesso em : 10 de nov. 2025

[5] *Scalable Private Set Intersection Based on OT Extension*, B.Pinkas, T.Schneider, Michael Zohner. Disponível em https://eprint.iacr.org/2016/930.pdf. Acesso em 12 de nov. 2025

Anexo A - Scripts

- naive.sh

```
#!/bin/bash
sizes=(5000 10000 20000 40000 80000)
outfile="results_naive.csv"

echo "Protocol,Dataset,Intersection,Time(s)" > $outfile

for n in "${sizes[@]}"; do
  echo "=== Protocolo 0 | ${n} registos ==="
  { time ./demo.exe -r 0 -p 0 -f psi_datasets/partyA_${n}.csv &>
server_${n}.log & } 2> time_tmp.txt
  SERVER_PID=$!

  output=$(./demo.exe -r 1 -p 0 -f psi_datasets/partyB_${n}.csv)
  inter=$(echo "$output" | grep -oP 'Found \K[0-9]+')
  runtime=$(grep real time_tmp.txt | awk '{print $2}' | tr -d 's')

  echo "0,${n},${inter},${runtime}" >> $outfile
  sudo fuser -k 7766/tcp 2>/dev/null

done

echo "Resultados guardados em $outfile"
```

- ttp.sh

```
#!/bin/bash
sizes=(5000 10000 20000 40000 80000)
outfile="results_serveraided.csv"

echo "Protocol,Dataset,Intersection,Time(s)" > $outfile

for n in "${sizes[@]}"; do
  echo "=== Protocolo 1 | ${n} registos ==="
  ./demo.exe -r 0 -p 1 -f README.md &
  SERVER_PID=$!

  { time ./demo.exe -r 1 -p 1 -f psi_datasets/partyA_${n}.csv & } 2>
time_tmp.txt &
  output=$(./demo.exe -r 1 -p 1 -f psi_datasets/partyB_${n}.csv)
  inter=$(echo "$output" | grep -oP 'Found \K[0-9]+')
  runtime=$(grep real time_tmp.txt | awk '{print $2}' | tr -d 's')

  echo "1,${n},${inter:-0},${runtime:-error}" >> $outfile

  sudo fuser -k 7766/tcp 2>/dev/null
  kill $SERVER_PID 2>/dev/null
done

echo "Resultados guardados em $outfile”
```

- dh.sh

```
#!/bin/bash
sizes=(5000 10000 20000 40000 80000)
outfile="results_diffie.csv"

echo "Protocol,Dataset,Intersection,Time(s)" > $outfile

for n in "${sizes[@]}"; do
  echo "=== Protocolo 2 | ${n} registos ==="
  { time ./demo.exe -r 0 -p 2 -f psi_datasets/partyA_${n}.csv &>
server_${n}.log & } 2> time_tmp.txt
  SERVER_PID=$!

  output=$(./demo.exe -r 1 -p 2 -f psi_datasets/partyB_${n}.csv)
  inter=$(echo "$output" | grep -oP 'Found \K[0-9]+')
  runtime=$(grep real time_tmp.txt | awk '{print $2}' | tr -d 's')

  echo "2,${n},${inter},${runtime}" >> $outfile
  sudo fuser -k 7766/tcp 2>/dev/null
done

echo "Resultados guardados em $outfile"
```

- ot.sh

```
#!/bin/bash
sizes=(5000 10000 20000 40000 80000)
outfile="results_ot.csv"

echo "Protocol,Dataset,Intersection,Time(s)" > $outfile

for n in "${sizes[@]}"; do
  echo "=== Protocolo 3 | ${n} registos ==="
  { time ./demo.exe -r 0 -p 3 -f psi_datasets/partyA_${n}.csv &>
server_${n}.log & } 2> time_tmp.txt
  SERVER_PID=$!

  output=$(./demo.exe -r 1 -p 3 -f psi_datasets/partyB_${n}.csv)
  inter=$(echo "$output" | grep -oP 'Found \K[0-9]+')
  runtime=$(grep real time_tmp.txt | awk '{print $2}' | tr -d 's')

  echo "3,${n},${inter},${runtime}" >> $outfile
  sudo fuser -k 7766/tcp 2>/dev/null
done

echo "Resultados guardados em $outfile"
```