INF-3315 Assignment 3 - Oblivious Transfer

Kevin Mathias Bergan

October 28, 2025

1 Introduction

In modern healthcare systems, the exchange of sensitive data must balance utility with strict privacy requirements. Oblivious Transfer (OT) is a fundamental cryptographic protocol that enables such privacy-preserving interactions. In a 1-out-of-8 OT scenario, a receiver can obtain exactly one message out of eight possible options, without revealing which one was chosen to the sender. At the same time, the receiver gains no information about the remaining seven messages. This dual privacy property makes OT a powerful building block for secure multi-party computation, privacy-preserving data sharing, and cryptographic protocols in sensitive domains such as healthcare and finance [Wik25].

In this assignment, a hospital needs to identify a blood donor with a specific blood type, but both privacy and confidentiality must be preserved. A third-party donor service stores a database of donors and their corresponding blood types, and when the hospital makes a request, it should only receive the name of one donor who matches the requested type. However, the hospital must not gain any information about donors with other blood types, ensuring that no extra or unnecessary personal information is revealed. At the same time, the donor service itself should not learn which blood type the hospital is searching for, preventing it from inferring patient conditions or sensitive hospital needs. Since there are eight possible blood types (A-, A+, B-, B+, O-, O+, AB-, AB+), the challenge is to allow the hospital to privately retrieve only the donor corresponding to one specific type, while keeping both the hospital's query and other donor data hidden.

2 Implementation

My implementation demonstrates a layered 1-out-of-2 Oblivious Transfer (OT) protocol that enables a receiver to securely obtain one out of eight possible messages from a sender, without revealing which message was chosen and without learning anything about the others. The construction is achieved by layering three 1-out-of-2 OTs, forming a binary tree structure that represents 8 possible outcomes (since $2^3 = 8$). Each layer corresponds to one bit of the receiver's selection index, and together they define the unique path through the three layers that leads to the desired message.

Here is an example of my implementation of an 1-out-of-8 oblivious transfer protocol:

1. The Receiver chooses blood type O+ (index = 5)
2. The Receiver generates 3 public keys and a commitment and sends these to the sender. (Keeping only the private key from the chosen blood type (index = 5).
3. The Sender prepares 8 messages: A-, A+, B-, B+, O-, O+, AB-, AB+.
4. The Sender also generates 8 ciphertexts from the public keys from the Receiver, as well as a commitment. This is sent to the receiver.
5. The receiver then uses the private key (in binary form) from earlier to choose the correct path down the binary tree to end up with the correct ciphertext to decrypt.

The protocol works by encoding the hospital's choice of blood type as three binary bits (since there are 8 possibilities), and then running three separate 1-out-of-2 Oblivious Transfers, one for each bit. The hospital generates three RSA key pairs and sends only the public keys to the donor service, along with a commitment to ensure they cannot later change them. For each of the three OT layers, the donor service encrypts two random symmetric keys: one corresponding to bit value 0 and one to bit value 1. The hospital, having the private keys, can decrypt exactly one symmetric key from each pair, uniquely determining a combination of three keys. Meanwhile, the donor service encrypts each donor record by XOR-masking it with the combined keys matching that record's index. When the hospital applies its recovered key combination, only the record matching the chosen blood type decrypts correctly—every other record remains unreadable. Throughout this process, the hospital obtains exactly one donor result without learning anything about others, and the donor service never learns which blood type was requested.

3 Discussion

My implementation provides strong privacy guarantees for both the hospital and the donor service. From the hospital's perspective, the use of layered 1-out-of-2 Oblivious Transfer ensures that it can only decrypt one combination of keys, and therefore only one donor record. All other records remain protected because each one is masked using a different key combination derived from the symmetric keys in each OT layer. Since the encryption is based on RSA with OAEP padding, the sender's encrypted keys are semantically secure, preventing the hospital from inferring anything about the unchosen keys. Meanwhile, the commitment over the public keys ensures that the hospital cannot change its OT request midway to try to learn multiple records, preserving the privacy of the donor database. Additionally, the donor service never learns which blood type was requested because it only sees generic RSA public keys and does not receive any data indicating which decryption path will be taken.

However, there are improvements that could enhance efficiency and security robustness. For example, this protocol currently uses fresh RSA key pairs for every request, which is computationally expensive. If this protocol instead was using elliptic-curve cryptography (ECC) or an OT extension protocol would significantly reduce overhead. Furthermore, the current scheme assumes honest-but-curious participants, meaning both sides are expected to follow the protocol correctly. It may also be beneficial to include mechanisms that allow auditing or rate-limiting requests to prevent a hospital from repeatedly querying to eventually learn all donor records.

If I were to implement for example Symmetric Oblivious Transfer (S-OT), the main improvement would be efficiency. Instead of generating fresh RSA key pairs and performing expensive public-key operations for every request, I could run a small number of base OTs once, and then use fast symmetric primitives (like AES-based PRGs and XOR operations) to derive the necessary keys for all future transfers. This reduces the computational cost dramatically and makes the protocol much faster and more scalable, especially when handling many queries or batching multiple transfers. It also allows me to precompute parts of the protocol offline, reducing response latency during real-time requests. However, I would need to ensure that the base OTs are kept secure, since the security of the extended OTs depends on them, and I may need to use a malicious-secure OT extension if I want protection against active adversaries rather than just honest-but-curious participants. Overall, S-OT would make the system far more efficient and practical while keeping the same privacy guarantees.

References

[Wik25] Wikipedia contributors. Oblivious transfer — Wikipedia, the free encyclopedia, 2025. [Online; accessed 28-October-2025].