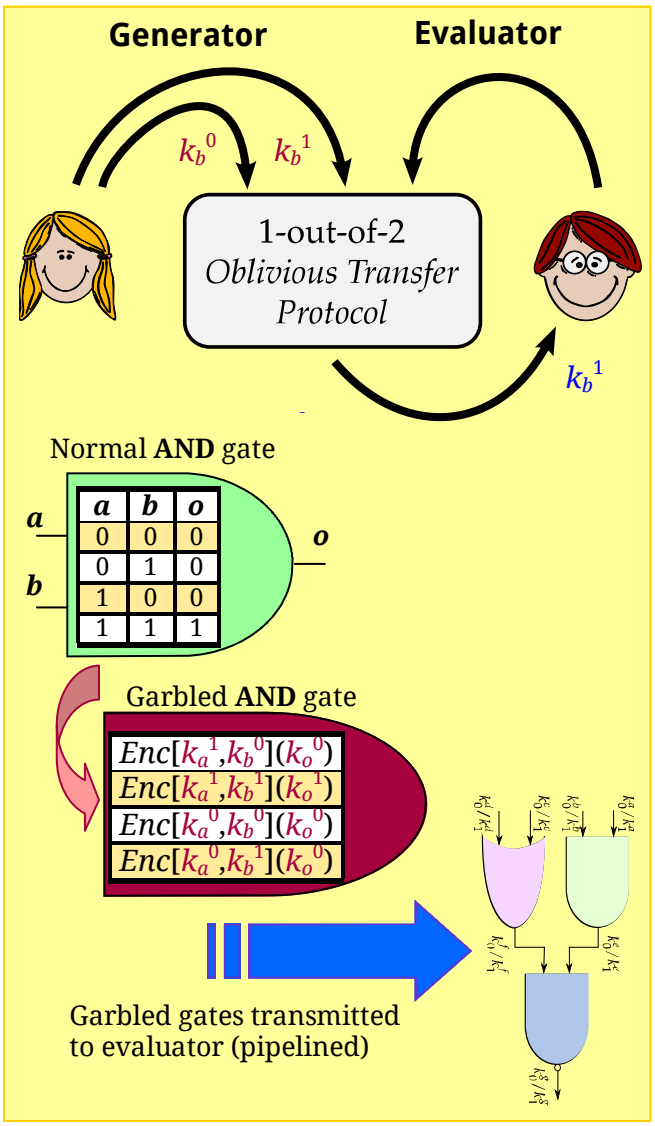
CommonContacts

Privacy-Preserving Shared Contact Computation

http://MightBeEvil.com/contacts

CommonContacts allows two users to collaboratively discover common entries in their address books *without disclosing any other information about their contacts*. The application uses a secure computation framework built using Yao's *garbled circuit* technique. All computation involving private data is performed on encrypted data so no information is released to the other party (other than what can be inferred from the result).

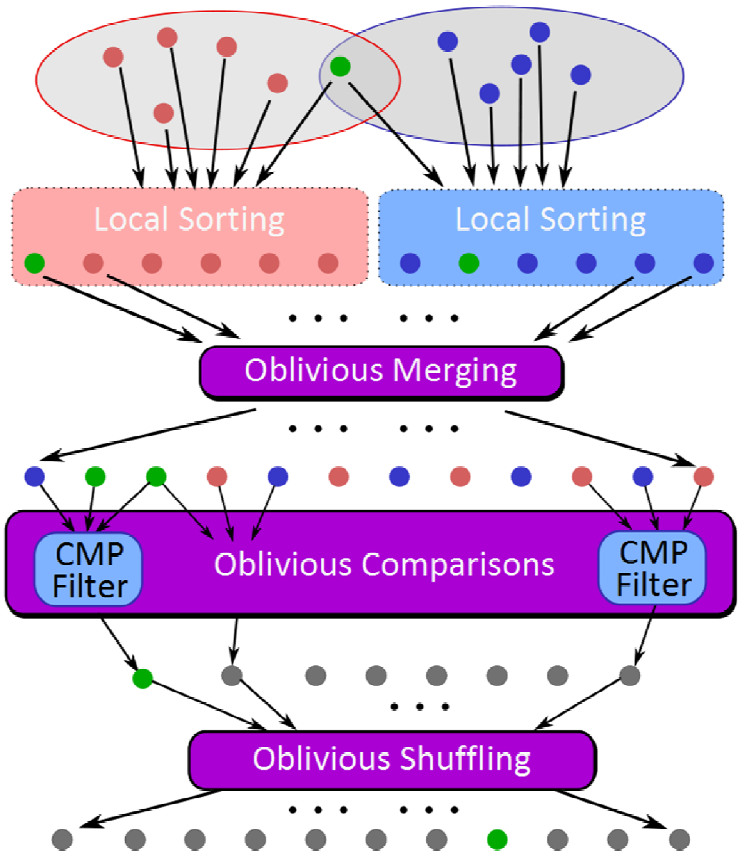


How it Works

Oblivious Transfer is a cryptographic technique that allows the Evaluator to obtain the encrypted values corresponding to his inputs (and only those values) without the Generator learning which values the Evaluator selected. For CommonContacts, the Evaluator selects the bits that represent his contact values, but the Generator learns nothing about which bits the Evaluator selected.

Yao's Garbled Circuits

Any discrete function can be implemented as a Boolean logic circuit. The circuit generator encrypts the circuit. Each logic gate is implemented as a lookup table, and the entries in the table are replaced with encrypted values. The evaluator evaluates a single path through the circuit, using the encrypted inputs obtained from the oblivious transfer. At the end of the protocol, the evaluator has the final encrypted output wires, which can be converted back to their semantic values to reveal the output without leaking any other information. Although this technique was originally of only theoretical interest, recent cryptographic advances and optimizations enable our fast garbled circuits framework to produce practical, privacy-preserving applications.



To compute the shared contacts, each party first produces its input which is a sorted set of the hashes of each entry in its address book. Then, the evaluator obtains the encrypted labels corresponding to his own set using oblivious transfer.

The **oblivious merging** circuit takes the two sorted sets as input and uses a *bitonic sorting* network to produce a single sorted list. This ensures that any matching elements are now adjacent.

The next step is to **obliviously compare** adjacent elements to find any matches (representing common contacts). This is done using a three-input comparison circuit that takes advantage of the property that the initial inputs are sets so there can only be one match from each triple.

It is not yet safe to reveal the result since the position where the match is found may leak information. We use a Waksman permutation network to **obliviously shuffle** the outputs, and then can safely reveal the result.

Privacy Implications

Assuming you trust our implementation and security assumptions, these applications leak no information about your private inputs other than what can be inferred from the result. This enables two individuals to safely learn their shared contacts without exposing any other entries in their address books. Secure computation offers the possibility for many useful privacy-preserving applications, but depends on users trusting and understanding the applications they run and install. We hope this application will both enable a useful private computation, as well as raising issues about how much normal applications expose user's sensitive data (including their address book).

For More Information

Yan Huang, David Evans, Jonathan Katz, and Lior Malka. *Faster Secure Two-Party Computation Using Garbled Circuits*, to appear in 20th USENIX Security Symposium, San Francisco, CA. 8-12 August 2011. [mightBeEvil.com/usenix11.pdf]

Yan Huang, David Evans, and Jonathan Katz. *Private Set Intersection: Are Garbled Circuits Better than Custom Protocols?*. [mightBeEvil.com/psi.pdf]

http://MightBeEvil.com

Yan Huang
Sang Koo

Peter Chapman
David Evans

UNIVERSITY of VIRGINIA