Private Web Search [PETS 2010]

• Yehuda Lindell and Erez Waisbard • Dept of Computer Science, Bar-Ilan, Israel

Abstract

Users' search terms to web search engines contain significant amounts of sensitive information and, as such, the aggregation and use of these terms constitutes a severe privacy breach. We propose a method that allows users to group together, randomly shuffle their search terms amongst themselves, and then query the search engine with the random term obtained.

We build on a previous solution by Castella-Roca et al. that achieves security in the semi-honest model. Our construction provides a way to achieve private web search in the malicious model, efficiently and practically, without having to resort to a full blown anonymous routing.

Private Web Search – Semi-Honest Case

Grouping is done by a trusted central server: Users register at the server and the server notifies them of the identities of their group members.

Users create a **joint ElGamal key**, encrypt their search term, and send it to everyone in their group.

A shuffle stage begins in which each member in turn **shuffles** (permutes) the vector of ciphertexts and **remasks** it (see below) so that its order cannot be linked to the previous vector. The last user sends the final permuted vector to everyone.

Users privately decrypt the permuted messages in a way so that each user obtains a single search query. The users submit the queries to the web search engine and send the results to everyone (so that the unknown initiator of the query can obtain the result).

ElGamal Encryption and Remasking

Secret key: x ***Public key:*** $y=g^x \bmod p$

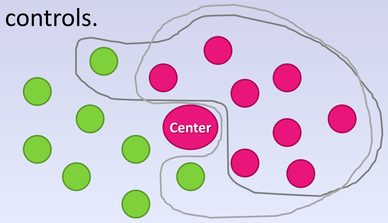
Encrypt M: *choose a random r and compute* $(g^r, y^r \cdot M)$
Decrypt (a,b): *Compute* $M = b/a^x$
Remask (a,b): *Choose a random r' and compute* $(a',b')=(a\cdot g^{r'}, b\cdot y^{r'})$

Remasking results in a different, unlinkable ciphertext of the same plaintext.
Observe that *Decrypt(a,b) = Decrypt(a',b') = M*

Moving to the Malicious model

The semi-honest protocol can be completely broken by a malicious adversary:

Grouping: A malicious center can always group a small number of honest users with malicious users that it controls.



Search results: All the parties learn the answer to the search query. This is also a privacy concern.

Attacks on the shuffle:

- The first party can replace all the search terms with copies of the one it is interested in. As a result everyone receives a decrypted copy of it at the last round and this goes unnoticed until it is too late.
- The last party to choose its share of the ElGamal key can choose it maliciously as a function of the values of all other parties. We have shown that it is possible to do this in a way that enables the last party to decrypt all values that are encrypted with the joint ElGamal key, without anyone noticing.
- The last party in the shuffle can use the initial vector of ciphertexts and not the permuted one it received from the previous party. In this case, all previous permutations are bypassed and the last party learns the source of each search term. This attack goes completely unnoticed.

Our Result – Privacy Against Malicious

Our private web search is composed of three stages, plus a one time initialization stage in which a certificate is created for a device. The stages are:

1. **Random grouping** of users with the help of a central server that acts as a bulletin board.
2. Users within a group permute their search queries using our **private shuffle** protocol that is secure in the presence of malicious adversaries.
3. **Privately sending the search results:** Using an encryption key that is sent along with the search term, the search result is encrypted prior to broadcasting it. This way only the initiator is able to decrypt the result.

Private Shuffle – Malicious Model

Assume a **PKI** is in place. Each party P_i holds a search term w_i.

Initialization Stage:

Each party creates two random ElGamal key pairs $(x, g^x), (y, g^y)$, and sends the public keys (g^x, g^y) with a Fiat-Shamir NIZKPOK of the secret keys (x,y) to all the other parties. All these messages are signed.

Next, joint ElGamal public-keys are computed as follows: $X = \Pi g^x = g^{\Sigma x}$ and $Y = \Pi g^y = g^{\Sigma y}$

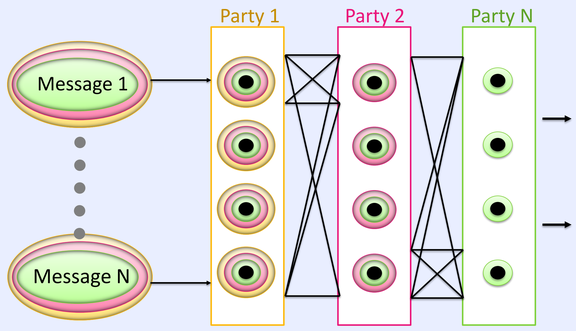
Each party P_i encrypts its w_i, first with X to obtain c_i and then with Y to obtain s_i. P_i then sends s_i to everyone.

Shuffle Stage:

Using any fixed ordering of the parties, party P_j carries out the following operations on the encrypted vector it received from P_{j-1}:

1. *It remasks all of the ciphertexts in the vector*
2. *It removes its "part" in the second encryption under Y (using its secret* y_j*)*
3. *It randomly permutes the encryption vector*

The last party sends the encrypted vector to everyone.



Verification Stage:

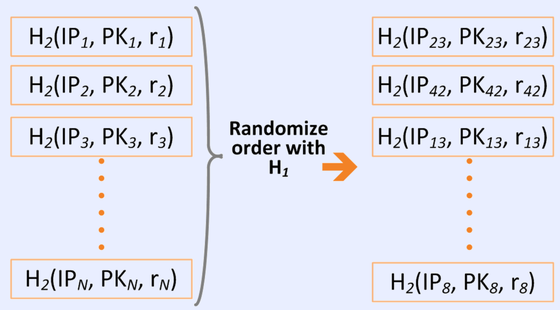
After all parties have removed their y_j parts, only the first layer of encryption under X remains. Every P_i then verifies that its encryption c_i is in the resulting vector. If yes, it sends a signed OK to everyone. This ensures that all parties participated in the shuffle stage and that no one tampered with the values during the process. (Note that no terms are revealed yet at this stage.)

Reveal stage:

If **all** parties OKed in the previous round, then every party P_i sends $g^{y_i \cdot r_j}$ to P_j so that it can recover w_i.

Random Grouping

- Let H_1 and H_2 be random oracles.
- All the users register at a central server S. Each user P_i registers with $H_2(IP_i, PK_i, r_i)$.
- The server S is the only one that knows the real IP addresses of the registered parties.
- After a short predefined time all the users compute H_1 of the list of all registered values.
- The output of H_1 is used to induce an order on the registered parties.
- Grouping is achieved by taking groups of size n according to the induced order.
- The server S notifies the users in the group of the IP addresses of their group members.
- Within a group all the users send their public keys and randomness that was used during registration, and all users verify that all the computations were carried out correctly. Everyone also verifies that grouping according to H_1 was carried out correctly.



Observe that the grouping order could have been computed by applying H_1 directly to (IP_i, PK_i, r_i). However, this would reveal all the IP addresses of all participants in all groups, which can lead to statistical attacks by the web search engine (see the paper).

We use H_2 in order to ensure that the IP addresses are revealed only to the members of the group.

References

J. Castell-Roca, A. Viejo and J. Herrera-Joancomarti. Preserving User's Privacy in Web Search Engines. *Computer Comm.*, 32(13-14):1541-1551, 2009.