Yao's millionaire problem

In 1982, Andrew Yao proposed the millionaire problem[1], which discussed how could two millionaires determine who is richer while keeping their actual wealth private.

1. secure multiparty computation

Secure multiparty computation(SMC), or secure computation, secure function evaluation(SFE) is an abstract of this kind of problems. SMC asks for protocols that enable several parties collaboratively compute a function without exposing their input.

More specifically, a set of parties, $P_1, P_2, \ldots, P_n$, each of whom has a input $x_i, 1 \le i \le n$, they want to evaluate $y = f(x_1, x_2, \ldots, x_n)$ while keeping x_i in private. Yao's millionaire problem is SMC with comparison as f.

2. Lin-Tzeng protocol

Lots of solutions have been proposed in literature to solve the millionaire problem. We only discusses the homomorphic encryption based solution proposed by Lin and Tzeng[2].

The sketch of Lin-Tzeng protocol[2] is, firstly encoding x, y(the millionaires' wealth) such that $S_x \cap S_y \neq \emptyset \Leftrightarrow x > y$, then the problem is how to determine the private set intersection of S_x, S_y, which is solved by a homomorphic encryption based subprotocol.

2.1. 0-encoding and 1-encoding

For $s = s_n s_{n-1} \ldots s_0 \in \{0,1\}^n$, 0-encoding of s is the set $S_s^0 = \{s_n s_{n-1} \ldots s_{i+1} 1 | s_i = 0, 1 \le i \le n\}$, invert the least significant bit of all prefix of s tailing 0. And 1-encoding of s is $S_s^1 = \{s_n s_{n-1} \ldots s_i | s_i = 1, 1 \le i \le n\}$, all prefix of s tailing 1. After encoding, $x > y$ if and only if $S_x^1 \cap S_y^0 \neq \emptyset$(we skip the proof here which is trivial).

2.2. multiplicative homomorphism

Homomorphism is a property provided by cryptosystems, within which, the operation on plaintext can be mapped into another operation on ciphertext, $E(x \times y) = E(x) \cdot E(y)$, so that we can outsource computation to an untrusted third party(cloud maybe). For instance, within textbook RSA, $E(xy) = (xy)^e \ mod \ N = x^e y^e \ mod \ N = E(x)E(y)$. We can let the cloud do the multiplication while keeping the input and output in private.

Additive homomorphism is quite the same. If a cryptosystem simultaneously provides additive and multiplicative homomorphism, we say it is fully homomorphic which is complete for all computable functions theoretically.

2.3. the protocol

1. Alice sends a matrix $T_{2\times n}$ to Bob, where $T[x_i, i] = E(1), T[\bar{x}_i, i] = E(r_i)$($r_i$ is random).
2. On receiving $T_{2\times n}$, Bob computes $c_t = T[t_n, n] \cdot T[t_{n-1}, n-1] \ldots \cdot T[t_i, i]$ for each $t = t_n t_{n-1} \ldots t_i \in S_y^0$, and chooses another $n - |S_y^0|$ random ciphertext forming a new set $\{c_1, c_2, \ldots, c_n\}$ which will be sent back to Alice after random permutation.
3. Alice decrypts all c_i, checks whether some of them are 1 which indicates $x > y$ and tells Bob the result.

If Bob responds $\{c_t\}$ directly without filling another $n - |S_y^0|$ random ciphertext, $\#0s$ of y is leaked.

Multiplicative homomorphic cipher in the protocol can be replaced by an additive homomorphic cipher, and use $E(0)$ other than $E(1)$ simultaneously.

2.4. correctness and security

Because of multiplicative homomorphism, if $D(c_t) = 1$, then $T[t_n, n], T[t_{n-1}, n-1], \ldots T[t_i, i]$ are all ciphertext of 1 with high probability, which means $x_n = t_n, x_{n-1} = t_{n-1}, \ldots, x_i = t_i$.

All messages observed by outside attackers are encrypted. Bob cannot differentiate $E(1)$ and $E(r_i)$ such that he gets no idea of x. With $\{c_t\}$, Alice also gains no information of y if she follows the protocol.

References

[1] A.C. Yao, Protocols for secure computations, in: Foundations of Computer Science, 1982, Sfcs'08. 23rd Annual Symposium on, IEEE, 1982: pp. 160–164.
[2] H.-Y. Lin, W.-G. Tzeng, An efficient solution to the millionaires' problem based on homomorphic encryption, in: International Conference on Applied Cryptography and Network Security, Springer, 2005: pp. 456–466.