Computazioni Sicure

23/02/2016

devim

Indice

Introduzione……...2

Oblivious transfer……..2

Garbled circuits……..2

Problematiche……4

Riferimenti……5

Introduzione

Una computazione sicura tra N partecipanti si verifica quando ciascuno di essi possiede dei dati privati che vogliono utilizzare per calcolare il valore di una funzione pur mantenendo i loro input segreti. Questa è una particolare applicazione del problema noto come *commitment scheme*. Un particolare sottoproblema, oggetto di interesse per questa ricerca, considera solamente due partecipanti: Alice e Bob. Esistono diversi protocolli ideati per tale scopo, i quali devono considerare diversi casi. In un caso, entrambi gli utenti potrebbero essere onesti e seguire fedelmente il protocollo, ma per scenari diversi e più interessanti abbiamo che uno o entrambi potrebbero essere semi-onesti o malevoli. Un utente semi-onesto è colui che segue il protocollo correttamente, ma che allo stesso tempo cerca di scoprire l'input dell'altro, mentre un utente disonesto può anche agire diversamente da quanto previsto dal protocollo. Importante è anche il concetto di *oblivious transfer* che verrà trattato nel paragrafo successivo.

Oblivious transfer

Con *oblivious transfer* (OT) si intende un tipo di protocollo pensato per trasmettere una di più parti di informazione che mantenga oscura (*oblivious*) la scelta fatta dal ricevitore. Questa è una caratteristica importante poiché è alla base di altri protocolli, di cui un esempio sarà visto nel seguito. Una particolare forma di *oblvious transfer* chiamata *1-out-of-2 OT,* basata su RSA, permette a due utenti, Alice e Bob, di comunicare in modo tale che Bob possa scegliere una tra due informazioni possedute da Alice senza che Alice conosca la scelta fatta e senza che Bob conosca l'altra informazione che non ha scelto [1]. Un possibile protocollo, sicuro solo contro avversari semi-onesti, prevede quanto segue:
Alice possiede due stringhe s_0 e s_1. Bob sceglie un indice $i \in \{0,1\}$ corrispondente all'informazione che desidera avere. In seguito genera una coppia chiave pubblica/privata e una seconda chiave pubblica, ma per la quale non ha la corrispondente chiave privata. Quindi Bob espone le due chiavi pubbliche che verranno usate da Alice per cifrare s_0 con una e s_1 con l'altra. Bob, possedendo solo una delle chiavi private, potrà decifrare esclusivamente una delle due informazioni. L'ordine con cui Bob espone le chiavi è importante per stabilire quale informazione intende ricevere.

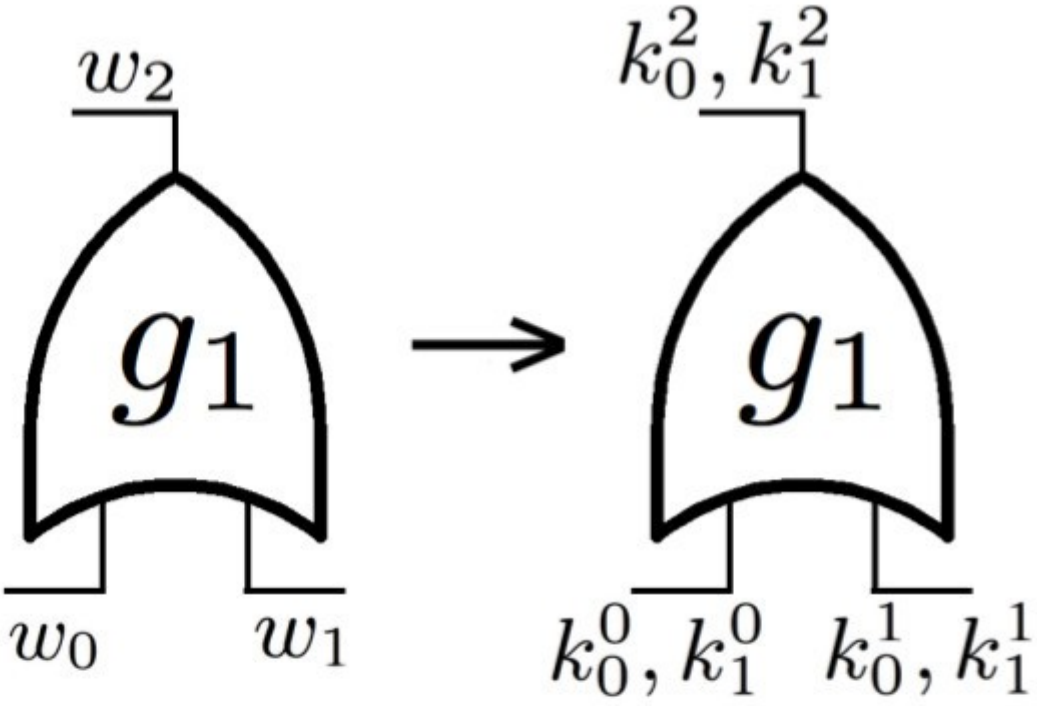
Garbled circuits

Il problema riguardante la computazione sicura fu introdotto da Andrew Yao nel 1982 [2], il quale definì un protocollo nato per risolvere il problema della computazione sicura tra due parti. Un chiaro esempio fornito da Yao stesso è il seguente:

> *"Due milionari desiderano conoscere chi è il più ricco; tuttavia, essi non vogliono lasciar fuoriuscire inavvertitamente qualsiasi informazione addizionale sul loro patrimonio. Come possono realizzare una tale conversazione?"*

Nel protocollo viene fatta l'assunzione semplificativa che non ci siano utenti malevoli, ma solo utenti onesti e/o semi-onesti. Si assume che le funzioni di hashing utilizzate impediscano di dedurre l'input esaminando il loro output. In particolare, si considera il fatto che le funzioni possono essere rappresentate da equivalenti funzioni booleane, quindi si sfruttano dei "circuiti booleani" formati da porte logiche, ma poiché sono cifrati vengono detti *garbled circuits*. Si noti che per ciascuna funzione di tempo polinomiale con input di dimensione fissata esiste un circuito booleano che la

rappresenta [3]. Si consideri, ad esempio, una singola porta OR (figura 1). Inizialmente Alice genera normalmente la sua tabella di verità. Successivamente crea una chiave per ogni possibile valore che può assumere ciascun input e output della porta. Quindi, nell'esempio, si avranno 6 chiavi, di cui 4 generate per gli input e 2 per l'output. Alice sfrutterà queste chiavi per cifrare ogni riga della tabella di verità e le rimescolerà con ordine casuale.



w_0	w_1	w_2
0	0	0
0	1	1
1	0	1
1	1	1

(a) Original Values

w_0	w_1	w_2	garbled value
k_0^0	k_1^0	k_2^0	$H(k_0^0 \| \| k_1^0 \| \| g_1) \oplus k_2^0$
k_0^0	k_1^1	k_2^1	$H(k_0^0 \| \| k_1^1 \| \| g_1) \oplus k_2^1$
k_0^1	k_1^0	k_2^1	$H(k_0^1 \| \| k_1^0 \| \| g_1) \oplus k_2^1$
k_0^1	k_1^1	k_2^1	$H(k_0^1 \| \| k_1^1 \| \| g_1) \oplus k_2^1$

(b) Garbled Values

Figura 1. Cifratura di una singola porta logica OR (fonte [4]). Con g_1 si indica l'identificativo della porta.

In questo modo Bob non potrà studiare il circuito per capire la funzione che rappresenta. Alice, allo stesso modo, può cifrare più porte logiche e combinarle tra loro per creare circuiti più complessi. L'unica informazione che è visibile da Bob è il risultato della computazione, perciò non è necessario cifrare la porta di output. Una volta generato il *garbled circuit*, Alice cifrerà il proprio input e invierà entrambe le informazioni a Bob. L'input cifrato consiste nella chiave associata al valore di input per una specifica entrata, quindi se Alice avesse un 1 per l'entrata w_0 userebbe $|0\rangle_+$. Alice ripeterà la procedura per tutti gli input che possiede relativi alle varie entrate del circuito. Bob a sua volta dovrà cifrare il proprio input, ma poiché non conosce le stringhe pseudo-casuali che rappresentano 0 e 1 dovrà intraprendere un *oblivious transfer* di tipo *1-out-of-2* con Alice. In seguito Bob avrà tutte le informazioni necessarie (gli input e il circuito cifrati) per il calcolo del risultato del

circuito. Per ciascuna porta di input, Bob cerca i valori corrispondenti al suo input cifrato e quello di Alice e li usa come chiavi per decifrare l'output presente nella tabella di verità. Nel particolare, con riferimento all'esempio della porta OR, Bob calcolerà l'hash della concatenazione del suo input con quello di Alice e con l'identificativo della porta stessa. Poiché l'operazione XOR è simmetrica, se lo si applica ad uno dei valori cifrati della tabella della verità con l'hash appena calcolato, si otterrà l'output (cifrato). Bob, non conoscendo a priori quale valore della tabella deve essere usato per l'operazione, dovrà necessariamente provarli tutti fino a che non ottiene un output valido. Gli output del circuito saranno bit in chiaro che verranno poi spediti ad Alice per completare il protocollo.

Problematiche

Il protocollo di Yao presenta diverse problematiche che lo rendono difficilmente applicabile nella pratica. Infatti il protocollo è pensato per essere sicuro contro utenti semi-onesti, ma rimane insicuro contro utenti malevoli. Nel tempo sono stati condotti diversi studi per migliorare tale protocollo, principalmente ci si è concentrati nello sviluppo di protocolli *OT* resistenti ad utenti malevoli, a garantire la generazione di un *garbled circuit* corretto e a prevenire la possibilità da parte di Alice di ottenere informazioni sull'input di Bob inviandogli valori corrotti del proprio input. Una questione aperta che rimane è come forzare Bob a comunicare il risultato finale ad Alice (*fairness*).
Negli anni sono stati fatti diversi studi sulla possibilità di realizzare un meccanismo di computazione sicura per sistemi quantistici, ma come sostenuto da Hoi-Kwong Lo [5] e, nel dicembre 2015, da un articolo di A. Broadbent e C. Schaffner [6], ciò risulta impossibile. Hoi-Kwong Lo dice che realizzare un *oblivious* transfer quantistico è impossibile, ma l'argomento è comunque oggetto di studi e infatti Li Yang, nel marzo del 2015, ha presentato una ricerca dove presenta un protocollo di questo tipo [7].

Riferimenti

[1] S. Even, O. Goldreich, and A. Lempel, "A Randomized Protocol for Signing Contracts", 1985, http://www.lix.polytechnique.fr/~catuscia/teaching/papers_and_books/SigningContracts.pdf

[2] Andrew Yao, “Protocols for secure computation”, http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=4568388

[3] O. Goldreich, S. Micali, A. Wigderson, “How to play any mental game”. In Proceedings of the nineteenth annual ACM symposium on Theory of computing, pagine 21–229. ACM, 1987.

[4] Peter Snyder, “Yao’s Garbled Circuits:Recent Directions and Implementations”, https://www.cs.uic.edu/pub/Bits/PeterSnyder/Peter_Snyder_-_Garbled_Circuits_WCP_2_column.pdf

[5] Hoi-Kwong Lo, “Insecurity of Quantum Secure Computations”, 1997, http://arxiv.org/abs/quant-ph/9611031

[6] Anne Broadbent e Christian Schaffner, “Quantum cryptography beyond quantum key distribution”, 2015, http://link.springer.com/article/10.1007/s10623-015-0157-4

[7] Li Yang, “Quantum oblivious transfer and bit commitment protocols based on two non-orthogonal states coding”, 2015, http://arxiv.org/abs/1306.5863