Round Complexity Lower Bound of ISC Protocol with $\mathcal{F}_{\mathsf{parpuz}}$ Model

Huijing Gong

December 11, 2014

1 Introduction

In the open, peer to peer networks where parties are not necessary to be authorized by any prior means of authentication, [4] proposed a interactive set consistency (ISC) protocol to realize nontrivial security over this type of scheme by defining precise bounds on computational power.

Their protocol relies on ISC properties, which could be used to establish public key infrastructure among parties. The ISC protocol involves two types of time-lock puzzle, $\mathcal{F}_{\mathrm{puz}}$ and $\mathcal{F}_{\mathrm{parpuz}}$. $\mathcal{F}_{\mathrm{parpuz}}$ is different by granting power to adversary to work parallelly with other corrupted parties when interacting with time-lock puzzle oracle.

In ISC protocol with $\mathcal{F}_{\mathrm{parpuz}}$ model, it must take at least $\mathsf{f}+1$ rounds for the protocol to tolerate f faults. This project is to find round complexity lower bound for the ISC protocol with $\mathcal{F}_{\mathrm{parpuz}}$ model.

2 ISC Protocol with $\mathcal{F}_{\mathsf{parpuz}}$ Model

To better understand this problem, we first introduce a *Byzantine General Problems* proposed by [3].

In the scenario of Byzantine General Problems, commander and generals are camped outside a enemy city. Commander send battle order, attack or withdraw, to generals and generals can send message to each other and decide their action. There are traitors involved to confuse others and people do not know who is the traitor. If Commander is the traitor, then Commander can send order to different generals. If some of the generals are traitors, they can send message to others and may convince them that Commander is the traitor. The goal of designing a good Byzantine protocol is to make honest generals agree on a same battle order; and if Commander is found out to be a traitor, then then generals will withdraw.

The scenario for reaching agreement among multiparties with no pre-existing setup is more tricky. Since when parties receive messages from other parties, the receiver has no idea where the message sent from. Moreover, if two message received from different rounds are sent from one party. However, a message sent by an honest party in any run will be received by all other parties by the end of that run.

In the network model without pre-existing setup, adversary can corrupt parties to behave arbitrarily, inject message into the network, change messages they relay, and send message to subset of the honest parties.

2.1 Interactive Set Consistency (ISC)

Interactive Set Consistency protocol provides a method to create a set of identities in replace of Public-Key Infrastructure, which enables parties to.conduct authenticated communication later.

Protocols that realize ISC must holds the following properties:

1. Each honest party P_i outputs a (multi)set S_i containing at most n values.
2. Each honest party output the same (multi)set S_i.
3. (Multi)set S_i contains inputs from all the honest parties.

Note here the agreed (multi)set S_i can be used as a list of public keys which enable honest parties to construct PKI.

2.2 $\mathcal{F}_{\mathtt{parpuz}}$ Model

Time-Lock Puzzle is in place of trusted setup assumption, which makes sure that each honest party has equal computational power and f corrupted parties cannot solve any faster than f honest parties per round. But in $\mathcal{F}_{\mathtt{parpuz}}$ model, adversary can runs sequentially faster by a factor of f.

The $\mathcal{F}_{\mathtt{parpuz}}$ model is designed in [4], which models the time-lock puzzle. In this model, each party to produce a puzzle solution independently each round and an adversary who controls f parties can solve f puzzles per round in total. The scheme is to solve cryptographic puzzle upon request and check the solution upon request in polynomial time.

$\mathcal{F}_{\mathtt{parpuz}}$ model is composed by two algorithms, *solve* and *check*:

In *solve* subroutine, $\mathcal{F}_{\mathtt{parpuz}}$ oracle maintains a table T. Each party P_i sends (solve, x_i) to $\mathcal{F}_{\mathtt{parpuz}}$ oracle. For $i = 1, ..., n$, $\mathcal{F}_{\mathtt{parpuz}}$ oracle first check if (x_i, h_i) has been store in T or not. If yes, then return h_i to P_i. Otherwise, uniformly generates a $h_i \in \{0,1\}^\lambda$ and return h_i to P_i and store (x_i, h_i) in T.

In *check* subroutine, each party P_i sends (check, (x_i^1, h_i^1), $((x_i^2, h_i^2)$, ...) to $\mathcal{F}_{\mathtt{parpuz}}$ model. $\mathcal{F}_{\mathtt{parpuz}}$ model returns $(b_i^1, b_i^2, ...)$. $b_i^j = 1$ if $(x_i^j, h_i^j) \in T$, $b_i^j = 0$, otherwise.

Note here In *solve* algorithm, each honest party is allowed to call $\mathcal{F}_{\mathtt{parpuz}}$ model only once per round. And for each round of honest party, all the solve

request must be sent before any honest party receives its solution. But each round of corrupted parties can call $\mathcal{F}_{\mathtt{parpuz}}$ model one after another in sequence up to f times.

2.3 Intuition of ISC Protocol with $\mathcal{F}_{\mathtt{parpuz}}$ Model

The generic sequence of this type of protocols committed by honest parties should be as following:

1. Each party sends solve-query to $\mathcal{F}_{\mathtt{parpuz}}$ model and receive the solution.
2. Each party computes a message to send.
3. Message sent from each party are delivered to all the other parties.
4. Each party sends a list of puzzle-solution pairs to $\mathcal{F}_{\mathtt{parpuz}}$ model for verification.

ISC Protocol with $\mathcal{F}_{\mathtt{parpuz}}$ model contains two phases in [4], *MIning Phase* and *Communication* Phase.

In *mining* phase, each correct party generate a chain of $\mathsf{O}(\mathsf{f}^2)$ puzzle solutions, for instance, Solve($\mathsf{pk_i}$, Solve($\mathsf{pk_i}$, Solve(Solve($\mathsf{pk_i}$, ø)))). Each correct party can create a valid puzzle chain for its own key but corrupt party only can create at most f puzzle chains before the protocol terminate.

In *communication* phase, each party publishes their chains and propagate the puzzle chain they received from others. In each round r, each party accepts a value if it has received a collection of r signatures on that value, the process then add its own signature to the collection and deliver it to the other processes. Signatures without associated puzzle chains are ignored. A correct party considers a public key '"valid" if it comes along with a puzzle chain containing the public key long enough

References

[1] Aguilera, Marcos Kawazoe and Toueg, Sam. *A simple bivalency proof that t-resilient consensus requires t+ 1 rounds.* Information Processing Letters. Volume 71.3 Page 155–158, 1999

[2] Dolev, Danny and Strong, H. Raymond. *Authenticated algorithms for Byzantine agreement.* SIAM Journal on Computing. Page 656–666, 1983.

[3] Fischer, Michael J and Lynch, Nancy A and Paterson, Michael S. *Impossibility of distributed consensus with one faulty process.* Journal of the ACM (JACM). Volume 32.2, Page 374–382, 1985.

[4] Katz, Jonathan and Miller, Andrew and Shi, Elaine. *Pseudonymous Secure Computation from Time-Lock Puzzles.* Cryptology ePrint Archive. no.857, 2014.