The Theory Behind Blockchains (Spring 19)
Recitation 3

Eliad Tsfadia

1 Byzantine Agreement

In class we saw a BA protocol against $t < n/2$ (PPT) corrupted parties that requires a setup phase and $t + 1$ rounds. Here we present a simpler BA protocol when $t < n/3$ that has only (expected) constant number of rounds. This protocol, however, assumes the existence of public random coins (beacon) that all parties agree on. In Section 2 we will hint how it is possible to generate this public beacon.

1.1 Rabin's Agreement Protocol (In the Random Beacon Model)

Assume there are $n = 3t + 1$ parties and at least $2t + 1$ of them are honest (the remaining t could be arbitrarily misbehaved or faulty). Initially, party i holds a bit $b_i \in \{0, 1\}$. The goal for them is to agree on a bit b_{final}. Required properties:

1. **Termination:** Eventually, every honest party decides some value.
2. **Agreement:** Every honest party must agree on the same value.
3. **Integrity:** If all honest parties proposed the same value b, then all of them must decide b.

Rabin's protocol (the algorithm for a honest party):

1. Set *vote* to the input bit ($vote = b_i$ for the i'th party).
2. Do until Break: (HW: determine when to break)
   (a) Send *vote* to all parties.
   (b) Receive votes from all other parties.
   (c) Set maj = the majority bit among all votes (including *vote*), and *count* = the number of occurrences of maj in all votes.
   (d) If $count \geq 2t + 1$: Set $vote = maj$.
   (e) If $count \leq 2t$: Set $vote = g\_bit$, where *g_bit* is a global coin-toss.
3. Output *vote*.

Analysis of Rabin's Protocol: Note that if all the honest parties have the same initial value, then they all set *vote* to this value in the first round. In all other cases, we show that with probability at least 1/2, all parties assign the same value to *vote* (Note that as soon as this happens, then from then on, $count \geq 2t+1$ for all parties, and so all parties will continue executing line (2d) in the algorithm).

There are two cases. (i) Some party sees $count \geq 2t+1$, and $maj = b$ for some $b \in \{0,1\}$. Since only t parties are faulty, we conclude that at least $t+1$ honest parties must have sent b as their value of *vote*. Thus, no other party will see both $count \geq 2t+1$ and $maj = 1-b$ in the same round. Hence, regardless of whether the other parties execute step (2d) or (2e) in the above algorithm, the probability is at least 1/2 that they will all set vote to b.

(ii) No honest party sees $t+1$. Then all of them execute step (2e), and with probability 1 set *vote* to the same value.

HW: Choose the break condition such that the protocol will satisfy:

1. Termination after constant number of rounds (in expectation).
2. Conditioned on termination, Agreement and Integrity holds.

2 Generating Public Random Beacon

In the following, we show how to generate public random beacon using Verifiable Secret Sharing scheme. We start by defining what is a Secret Sharing Scheme.

2.1 Secret Sharing Scheme (SSS)

Secret sharing is a method for distributing a secret amongst a group of participants. In a k-out-of-n secret-sharing scheme (SSS), we have n participants and a dealer. The dealer split a secret s into shares, where each of the n participants gets one of the shares. In order to reconstruct the secret, the participants must combine at least k of the shares, and any subset of $k-1$ shares does not leak any information about the share s.

Easy case (n-out-of-n SSS): Assume the dealer has a secret $s \in \{0,1\}^\ell$. An n-out-of-n scheme can be implemented as follows: The dealer chooses $n-1$ uniform strings $r_1, \ldots, r_{n-1} \in \{0,1\}^\ell$ and for every $i \in [n-1]$ it sends $\text{share}_i = r_i$ to the i'th party, and to the last party it sends $\text{share}_n = s \oplus r_1 \oplus \ldots \oplus r_{n-1}$ (the bitwise XOR of all strings). First, note that all n parties can reconstruct s since $s = \oplus_{i=1}^{n} \text{share}_i$. Second, note that any subset of $n-1$ shares is distributed uniformly and therefore does leak any information about s.

2.2 Verifiable Secret Sharing (VSS)

An k-out-of-n VSS scheme is an k-out-of-n SSS with the following two additional properties:

1. **Detecting a faulty dealer:** A faulty dealer might send invalid shares, i.e. shares that doesn't reconstruct to the same secret. We require that any subset of $k' \geq k$ shares reconstruct to the same secret. Otherwise, parties will be able to detect this faulty behavior.
2. **Detecting a faulty party:** A faulty party might send an incorrect share as part of the reconstruction process. We require that any such behavior will also be detected by the other parties.

2.3 Generating Common Random Bits using VSS

Assume that the number of faulty parties $t < n/2$. Each party $i \in [n]$ chooses a random string $s_i \in \{0,1\}^\ell$ and runs a $(t+1)$-out-of-n VSS protocol with the other parties (including itself as one

of them) where it plays the role of the dealer. After that, the parties reconstruct all $s_1, \ldots, s_n$ and set the common string to be $s = \oplus_{i=1}^{n} s_i$ (the bitwise XOR of all strings). If the i'th party behaved faulty as a dealer in its VSS protocol, then the honest parties will detect it and will set $s_i = 0$. Otherwise (i.e., s_i is shared correctly), the reconstruction of s_i always succeed since there are at least $t + 1$ honest parties. In addition, the output string is uniform since the faulty parties chose their strings (and therefore, committed to them) before knowing what are the strings of the honest parties.

2.4 Constructing VSS

Will be posted later.