RELIABILITY OF DISTRIBUTED SYSTEMS

Mike Greenbaum and David Ponarovsky

February 26, 2022

First we will create a broadcast algorithm for later use. Assume Broadcaster wants to send message v. The algorithm is:

Algorithm 1: broadcaster, j'th party

1 send $\langle broadcast, v\rangle$ to all parties

Algorithm 2: i'th party broadcast with j'th party as broadcaster

1 **if** *received first proposal* $\langle broadcast, v\rangle$ *from broadcaster* **then**
2 | send $\langle echo, v\rangle$ to all parties
3 **end**
4 **if** *received* $\langle echo, v\rangle$ *from* $n-2f$ *distinct parties* **then**
5 | **if** *didn't send* $\langle echo, v\rangle$ *yet* **then**
6 | | send $\langle echo, v\rangle$ to all parties
7 | **end**
8 **end**
9 **if** *received* $\langle echo, v\rangle$ *from* $n-f$ *distinct parties* **then**
10 | accept message v and terminate.
11 **end**

Broadcast validity If the broadcaster is non-faulty then, after at most 2Δ time, all non-faulty parties will output the broadcaster's input.

Proof:

If a non-faulty party started the protocol. Then after at most Δ time, every non-faulty party heard $\langle broadcast, v\rangle$. Therefore, after at most Δ, every non-faulty party sends $\langle echo, v\rangle$. After at most 2Δ, each non-faulty party heard $\langle echo, v\rangle$ from every non-faulty party. Therefore, after at most 2Δ, every non-faulty party heard at least $n-f$ $\langle echo, v\rangle$ and then they agree on v and terminate.

Broadcast agreement If some non-faulty party outputs a value then, after at most 2Δ, all non-faulty parties will output the same value.

Proof:

First Notice that every non-faulty party sends $\langle echo, v\rangle$ before it accepts via step 1 or 2.

If a non-faulty party outputs v then it heard at least $n-f$ $\langle echo, v\rangle$. Then it heard at least $n-2f$ non-faulty parties that sent $\langle echo, v\rangle$.

Assume by contradiction that a non-faulty party outputs $v' \neq v$. Therefore, by the same argument, it heard at least $n-2f$ non-faulty parties that sent $\langle echo, v\rangle$. Therefore, at least $n-2f$ non-faulty parties heard $\langle broadcast, v\rangle$ first and at least $n-2f$ non-faulty parties heard $\langle broadcast, v'\rangle$ first. By quorum, there is a non-faulty party that heard $\langle broadcast, v\rangle$ first and $\langle broadcast, v'\rangle$ first. This can't be because one of them arrives before the other. Therefore, every non-faulty party will output the same value

Due to the fact that a non-faulty party heard at least $n-2f$ non-faulty parties that sent $\langle echo, v\rangle$, therefore, after at most Δ every non-faulty party will hear $n-2f$ $\langle echo, v\rangle$ and will send $\langle echo, v\rangle$. Therefore, after at most 2Δ, every non-faulty party will hear from every non-faulty party $\langle echo, v\rangle$ and then every non-faulty party will accept v.

Adding Prefix to broadcast Notice that we can change broadcast to support broadcasting with prefix by sending echo only on the first message with a given prefix.

Now we will define a VSS protocol, let r be a prime number:

Algorithm 3: i'th party vss with j'th party as dealer

Round 1:
if *i is the dealer* **then**
  Sample $a_0, \ldots, a_{f^2} \sim Uni\{0, r\}$
  Define $p(x,y) = \sum_{k=0}^{f} \sum_{l=0}^{f} a_{k \cdot f + l} \cdot x^k \cdot y^l$
  Define $f_k(x) = p(x,k)$ and $g_k(y) = p(k,y)$.
  send $\langle f_k, g_k \rangle$ to party k, $\forall 1 \le k \le n$.
end
Round 2:
if *received $\langle f_i, g_i \rangle$ from dealer* **then**
  send $\langle f_i(k), g_i(k) \rangle$ to party k, $\forall 1 \le k \le n$.
end
Round 3:
wait 2Δ
for *party k, $\forall 1 \le k \le n$* **do**
  Denote the heard value of the party with u_k, v_k
  if *didn't hear a value from party k or $u_k \neq g_i(k)$ or $v_k \neq f_i(k)$* **then**
    broadcast $\langle complaint, i, k, f_i(k), g_i(k) \rangle$ with prefix *complaint* and i as the broadcaster.
  end
  wait 2Δ
  if *no broadcasts of complaint heard* **then**
    broadcast $\langle good \rangle$ with prefix *good* and i as the broadcaster
    go to Round 6
  end
end
Round 4:
if *i is the dealer* **then**
  Denote the complaints heard as a list of $\langle complaint, l, k, u, v \rangle$.
  lst=[]
  if $f_l(k) \neq u \vee g_l(k) \neq v$ **then**
    add $\langle l, f_l, g_l \rangle$ to the lst
  end
  broadcast $\langle reveal, lst \rangle$ with prefix *reveal* and i as the broadcaster.
end
Round 5:
wait 2Δ
for *broadcast $\langle complaint, k, l, u_1, v_1 \rangle$ heard in Round 3* **do**
  if *heard broadcast $\langle complaint, l, k, u_2, v_2 \rangle$ and didn't hear in dealer's broadcast $\langle reveal, l \rangle$ and not $\langle reveal, k \rangle$* **then**
    go to Round 6.
  end
end
for *reveal message $\langle reveal, l, f_l, g_l \rangle$ in revealed lst from dealer* **do**
  if $i == l$ **then**
    go to Round 6.
  end
  if $f_l(i) \neq g_i(l) \vee g_l(i) \neq f_i(l)$ **then**
    go to Round 6.
  end
end
broadcast $\langle good \rangle$ with prefix *good* and i as the broadcaster.
Round 6:
wait 10Δ
if *at least n-f parties broadcast-ed $\langle good \rangle$* **then**
  store $f_i(0)$
end

VSS liveness If a dealer started the protocol, the protocol will finish in a constant amount of time

Proof: Notice that we wait at most a constant amount of time in each round and therefore the amount of time the protocol runs is constant (up to a factor of Δ).

VSS honesty If a honest dealer started the protocol, then all honest parties will store the polynomial calculated by the dealer.

Proof: Notice that the complaints that contradict each other will be between an honest party and a faulty party. The dealer will reveal the faulty party's polynomial and therefore all honest parties will agree with the reveal. Therefore all honest parties will broadcast $\langle good \rangle$ and therefore all honest parties will store $f_i(0)$.

VSS hiding If a honest dealer started the protocol, then the faulty parties can't reconstruct the polynomial

Proof: Notice that the faulty parties will hear at most f polynomials which needs $f+1$ to reconstruct. Therefore, as stated in class, the faulty parties will see a uniform distribution of values and won't know the value until a honest party will reveal $f_i(0)$.

VSS agreement If a faulty dealer started the protocol and some honest party stored $f_i(0)$ then at least $n-2f$ points heard by honest parties fall on the same polynomial of degree f.

Proof: If a honest party stored $f_i(0)$, then it heard at least $n-f$ $\langle good \rangle$. Therefore, at least $n-f-f \geq f+1$ of the broadcasts were made by honest parties. Therefore, at least $n-2f$ parties agreed on their polynomials. Therefore, as stated in class, there exists only a single polynomial of degree f which agrees on $f+1$ two-variable polynomial of degree at most f. Therefore, the points stored by the $n-2f$ honest parties all fall on the same polynomial of degree f.

VSS correctness If a honest party stores a value then all honest parties store a value after at most 2Δ

Proof: If a honest party stored $f_i(0)$, then it heard at least $n-f$ $\langle good \rangle$. Therefore, due to the correctness of the broadcast, after at most 2Δ all honest parties will store their polynomial value.

Adding Prefix to VSS Again, to support this protocol to run on more than one dealer at the same time, we add prefix $dealer_j$ to every message of the VSS when j is the dealer.

Algorithm 4: i'th party random beacon

if *call getRand* **then**
    run VSS protocol and i as the dealer, allow to run only once
end
if *stored at least $f+1$ values with VSS* **then**
    wait 2Δ (for all honest parties to also store $f+1$ values)
    broadcast stored values $\langle dealer_j, f_i(0) \rangle$ with prefix $dealer_j$ and i as broadcaster.
    wait $2\Delta + 2\Delta$
    randomSeed $= 0$
    for *heard at least $n-f$ different parties with $\langle dealer_j, * \rangle$* **do**
        p = restore polynomial with heard values
        $randomSeed += p(0) \mod r$
    end
    output *randomSeed*
end

Correctness: all honest parties see the same value of the beacon

Proof:

Due to the correctness of the broadcast, all honest parties will hear the same messages. Furthermore due to VSS agreement, for each dealer heard, there will be at least $n-2f = 3f+1$ that fall upon the same polynomial.

Notice that we can reconstruct the polynomial due to the fact that with $5f+1$ values, there can be only one group of $3f+1$ that fall on the same polynomial. Assume by contradiction that it is not the case, therefore exists $v_1 \ldots, v_{3f+1}$ and $u_1, \ldots, u_{3f+1}$ that fall on different polynomials. Denote the groups of points with A, B We know that

$$2 \cdot (3f+1) - |A \cap B| = |A| + |B| - |A \cap B| = |A \cup B| \leq 5f+1 \implies |A \cap B| \geq f+1$$

Therefore, $|A \cap B| \geq f+1$, which means they agree on $f+1$ points. However, $f+1$ points define a single polynomial of degree f, contradiction.

Therefore, after we explained why the procedure is allowed, notice that all honest parties hear the same messages and reconstruct the same polynomials. Therefore, due to the deterministic of the code, all honest parties will output the same value of the beacon.

Liveness: if $f+1$ honest parties call getRand then a value is returned after a constant number of rounds

Proof: Notice that VSS takes a constant amount of time as proven in VSS liveness. Therefore, due to VSS honesty, after the $f+1$ honest party calls getRand, each honest party will store at least $f+1$ values after a constant amount of rounds. Then, notice that we enter line 4, from which we finish the protocol in 6Δ. Therefore, after $f+1$ honest parties call getRand then a value is returned after a constant number of rounds

Unpredictability: if no honest party calls getRand, then the adversary has no knowledge of the beacon value
Proof:

Notice that if no honest party calls getRand, no honest party will store $f+1$ values in the VSS. Therefore, no honest party will enter the restoration phase.

Due to the VSS hiding, the adversary doesn't know the value of the honest dealers that participate in getRand.

The adversary can't affect the calculated value in the restoration phase (only values agreed in VSS are allowed) and doesn't know the values of the honest dealers. Therefore, the adversary has to choose a value r before he knows the honest dealers values and participate in the VSS himself. Therefore, the adversary is bound to a value r.

Due to the fact that at least one honest dealer must complete the VSS before the restoration, the outputed value will be r + a random, uniform and independent value. Therefore, the final result will be a random, uniform and independent value. Therefore, if no honest party calls getRand, then the adversary has no knowledge of the beacon value.

Final Note: in the VSS, we allowed to pass only random values as the input, but we could easilty extend VSS to get input and use it instead of the coefficient for $x = 0, y = 0$ and it will be the recovered secret.