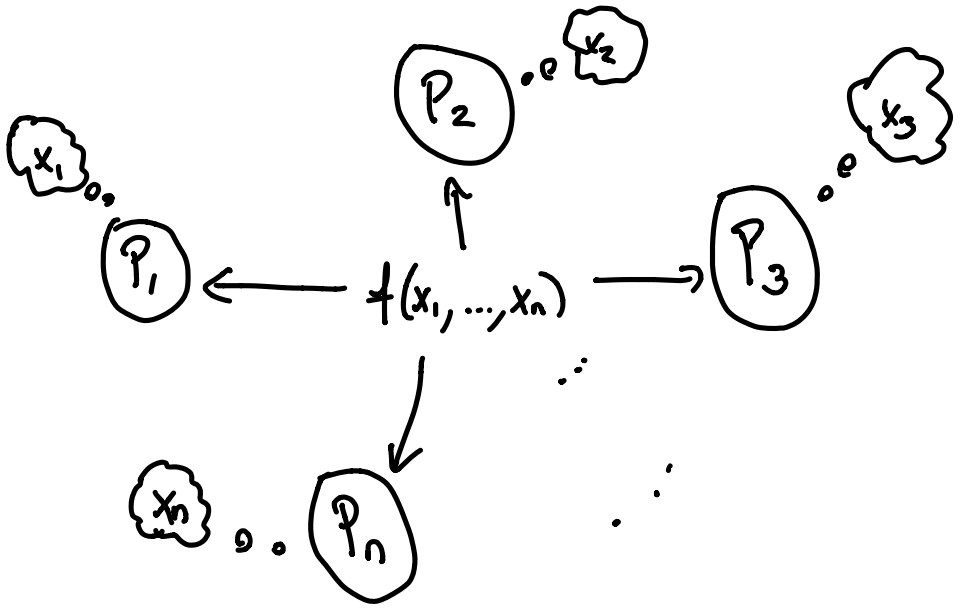
BGW Honest-But-Curious MPC



Want all P_i to learn $f(x_1, \ldots, x_n)$ without revealing secret inputs x_i.

Assumption: Players can communicate via pairwise channels (no broadcast).

Security Models:

Honest-But-Curious: Models "wire taps"

Adv chooses players to corrupt, gets to listen to all messages they receive + see their secret input

Corrupted players still follow protocol

Malicious: Corrupted players can behave arbitrarily. Players commit to inputs so computation can continue even if they go offline / deviate.

t-resilience: Even if Adv corrupts up to t players, cannot learn input of other players (beyond what is implied by the output f).

STATIC CORRUPTIONS: CORRUPTED PLAYERS CHOSEN AT START OF PROTOCOL

ADAPTIVE CORRUPTIONS: CAN CHOOSE PLAYERS TO CORRUPT DURING PROTOCOL AT ANY TIME BASED ON MESSAGES IN THE PROTOCOL

THM [BGW]:

$\exists$ PROTOCOL FOR $t < \frac{n}{3}$ RESILIENT MPC FOR MALICIOUS CORRUPTIONS (WITH PAIRWISE COMMUNICATION)

THM [RABIN, BEN-OR]:

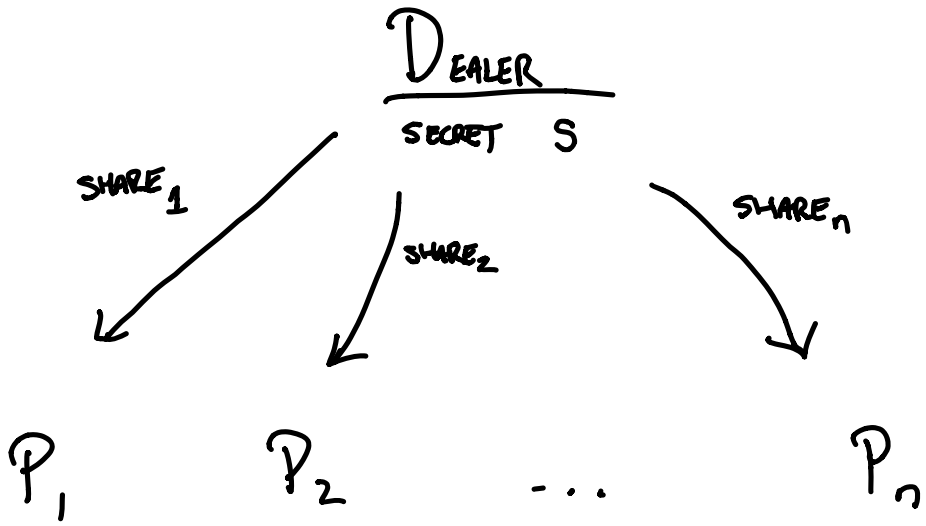
WITH BROADCAST, CAN DO $t < \frac{n}{2}$

How do we formalize security? Simulator!

For any t corrupted players, there exists a simulator Sim which, given the output $f(x_1, \ldots, x_n)$, simulates the messages from honest players to corrupted players.

BUILDING BLOCK: SHAMIR SECRET-SHARING:

(t,n) SECRET SHARING:



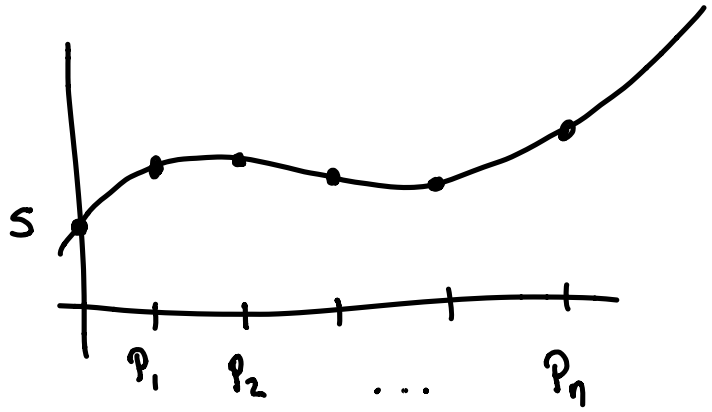
GOAL: ANY SET OF t PLAYERS CANNOT LEARN ANYTHING ABOUT 0

ANY SET OF $t+1$ PLAYERS CAN RECONSTRUCT S

CONSTRUCTION: D PICKS RANDOM COEFFICIENTS $a_1, \ldots, a_t$ & SETS POLYNOMIAL

$$p(x) := s + a_1 x + a_2 x^2 + \cdots + a_t x^t$$

$$\text{SHARE}_i = p(i)$$



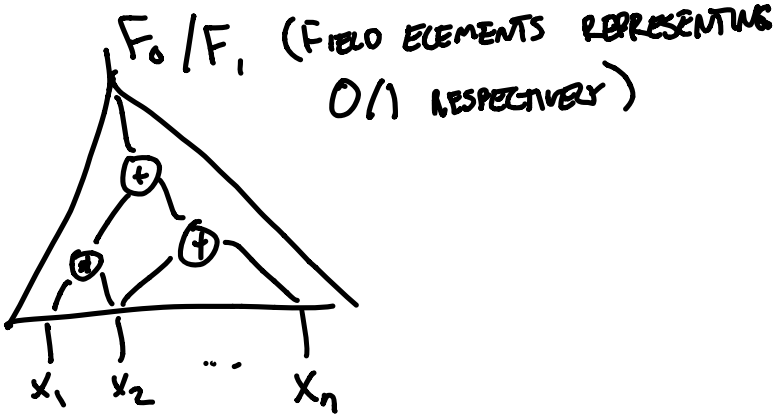
CRUCIAL FACT: GIVEN $t+1$ POINTS OF A DEGREE t POLYNOMIAL, CAN RECONSTRUCT ENTIRE POLYNOMIAL USING LAGRANGE INTERPOLATION

GIVEN $\leq t$ POINTS, GIVES NO INFORMATION ABOUT POLYNOMIAL

THUS: ANY $t+1$ PLAYERS CAN RECONSTRUCT p, COMPUTE $p(0)=s$. ANY $\leq t$ PLAYERS KNOW NOTHING ABOUT $p(0)$.

BACK TO BGW:

(1) REPRESENT f AS ARITHMETIC CIRCUIT:



(2) EACH P_i ACTS AS DEALER WITH SECRET X_i & DISTRIBUTES $(\frac{n}{3}, n)$ SHAMIR SECRET-SHARES OF X_i TO ALL PLAYERS (INCLUDING HIMSELF)

(3) FOR EACH GATE, COMPUTE SHARES OF OUTPUT WIRE GIVEN SHARES OF INPUT WIRES.

MUST SHOW HOW TO DO THIS FOR (+) & (*) GATES, THEN THIS COMPLETES THE PROTOCOL.

(+) : GIVEN $p_0(x) = s_0 + a_1 x + \ldots + a_t x^t$ +

$p_1(x) = s_1 + b_1 x + \ldots + b_t x^t$

P_i HOLDS $p_0(i)$ & $p_1(i)$

NEED PLAYERS TO HOLD SHARES OF SOME NEW POLYNOMIAL $p_{0+1}(x) = (s_0+s_1) + c_1 x + \ldots + c_t x^t$

FOR RANDOM c_i

<u>IDEA</u>: ADD SHARES $p_0(i) + p_1(i) \to p_{0+1}(i)$

ALMOST WORKS! c_i ARE NOT QUITE RANDOM. MUST JOINTLY CREATE RANDOM POLYNOMIAL $q(x) = 0 + d_1 x + \ldots + d_t x^t$ + ADD $q(i) + p_0(i) + p_1(i)$. THEN RANDOM COEFFICIENTS

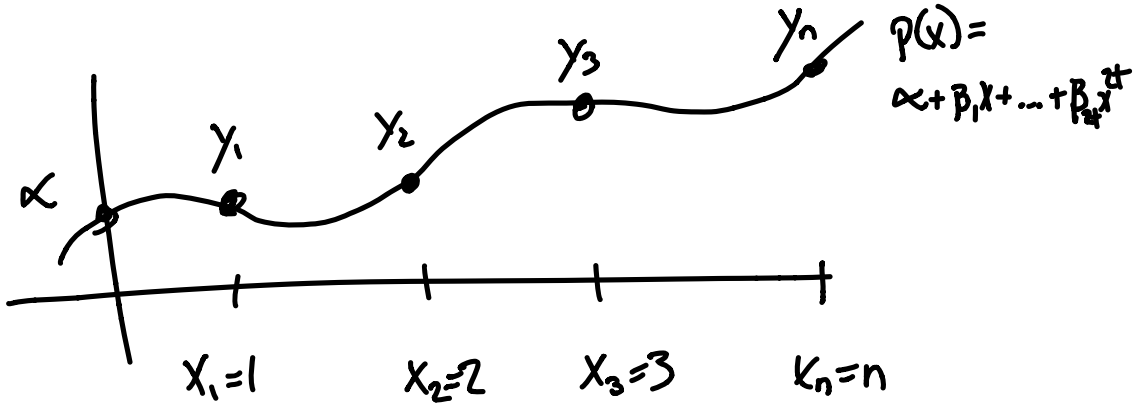
(*): IDEA: DO SAME AS (+),

$$P_0(i) * P_1(i) \rightarrow P_{0*1}(i)$$

PROBLEM: NEW P_{0*1} IS NOT RANDOM (SAME SOLUTION AS (+))

PROBLEM: NEW P_{0*1} IS DEGREE $2t$, NOT DEGREE t.

HOW DO WE REDUCE THE DEGREE?



LAGRANGE INTERPOLATION:

$$\alpha = L_1 \cdot y_1 + L_2 y_2 + \ldots + L_{2t+1} y_{2t+1}$$

PUBLIC, DEPEND ONLY ON x_i AND DEGREE $2t$

How does this help degree reduction?

P_i COMPUTES $y_i = p_0(i) * p_1(i)$

COMPUTES $L_i \cdot y_i$

SECRET SHARES NEW POLYNOMIAL

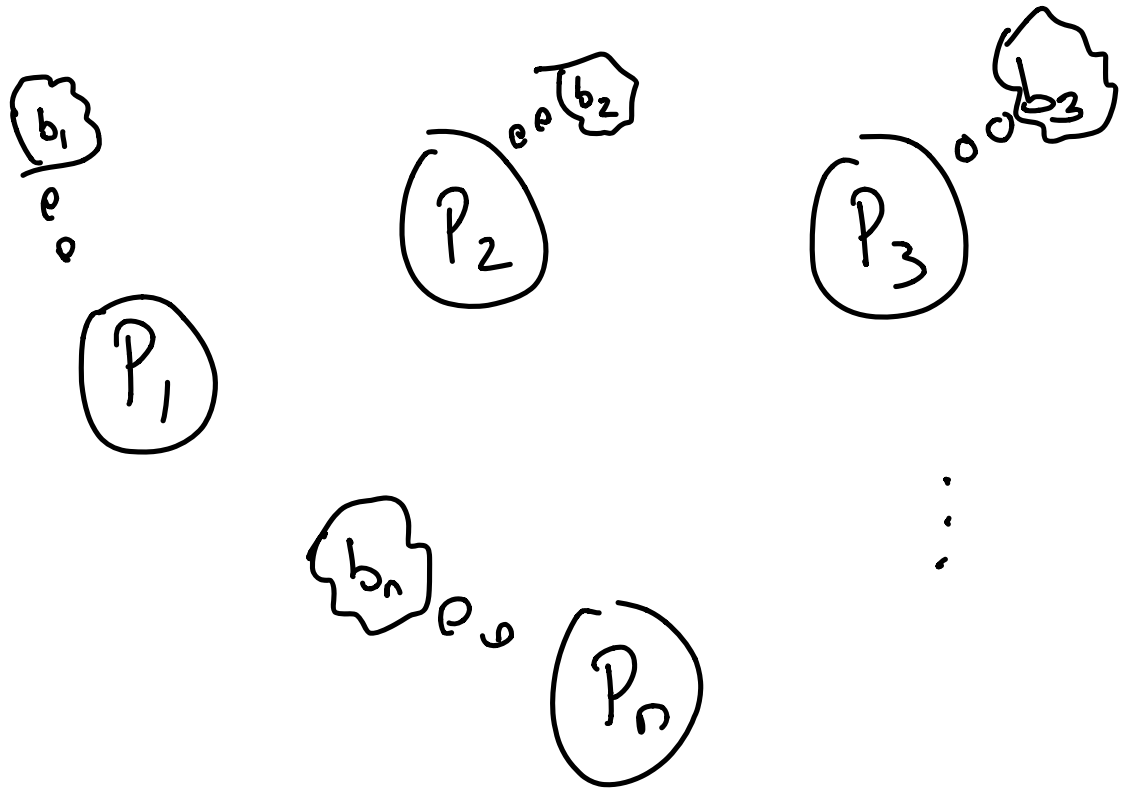
$$W_i(x) = L_i \cdot y_i + a_1 x + \ldots + a_t x^t$$

PLAYERS ADD ALL W_i SHARES TO GET SHARES OF $\propto$

$$W(x) = \overbrace{(L_1 \cdot y_1 + \ldots + L_{2t+1} \cdot y_{2t+1})} + \lambda_1 x + \ldots + \lambda_t x^t$$

WITH RE-RANDOMIZATION, THIS IS P_{OWF}!

Byzantine Agreement :



- EACH "GENERAL" WANTS TO EITHER ATTACK ($b_i = 1$) OR RETREAT ($b_i = 0$)
- CAN ONLY COMMUNICATE VIA PAIRWISE CHANNELS
- IF GENERALS ALL CHOOSE SAME ACTION (ATTACK OR RETREAT) THEY WILL BE SUCCESSFUL.

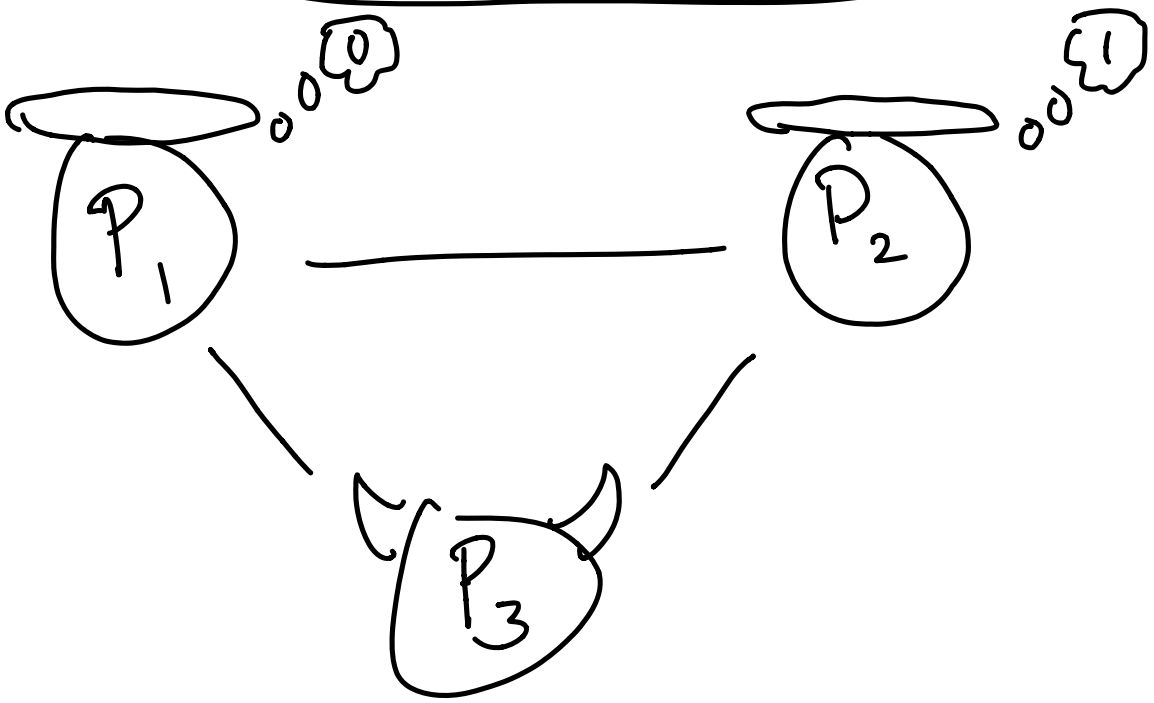
GOAL:

(1) IF ALL HONEST PLAYERS HAVE SAME INPUT $b = b_1 = \ldots = b_n$, ALL HONEST PLAYERS SHOULD OUTPUT b

(2) IF HONEST PLAYERS HAVE DIFFERENT VALUES, MUST STILL OUTPUT SAME VALUE (EITHER 0 OR 1)

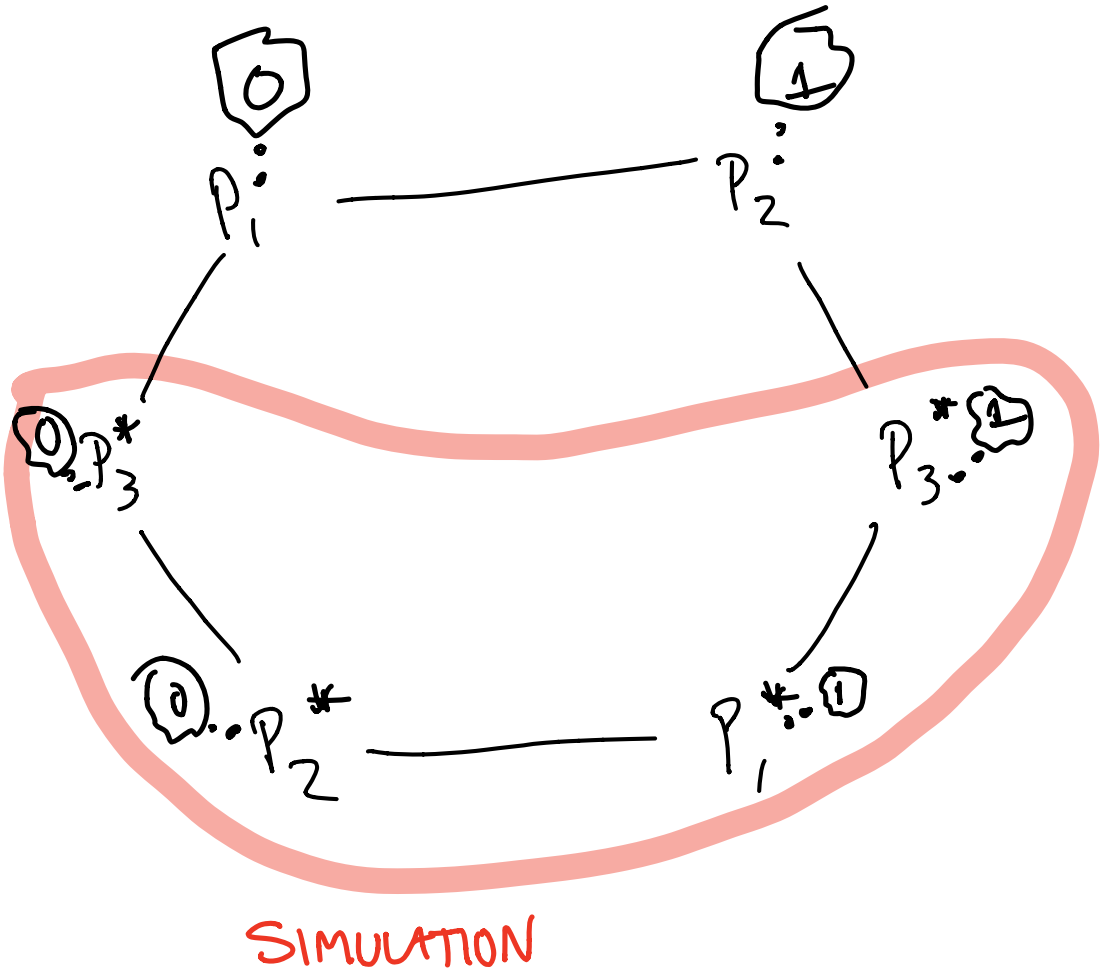
(3) UP TO $t < \frac{n}{3}$ GENERALS ARE CORRUPTED.

PROOF OF IMPOSSIBILITY FOR $n=3$



SUPPOSE $\exists$ PROTOCOL FOR $t=\frac{n}{3}$ CORRUPTIONS.

P_3 SIMULATES A COPY OF THE PROTOCOL IN HIS HEAD, KNOWING P_1 & P_2'S INPUTS


0
1
p_1
p_2
0
p_3^*
1
p_3^*
0
p_2^*
p_1^*
1
SIMULATION

P_3^* HONESTLY EXECUTES CODE OF PROTOCOL.

SINCE P_2 & P_3^* AGREE ON PRIVATE BIT 1,

P_2 MUST OUTPUT 1, SINCE BOTH ARE HONEST &

P_1 MAY BE DISHONEST. SIMILARLY, P_1 MUST

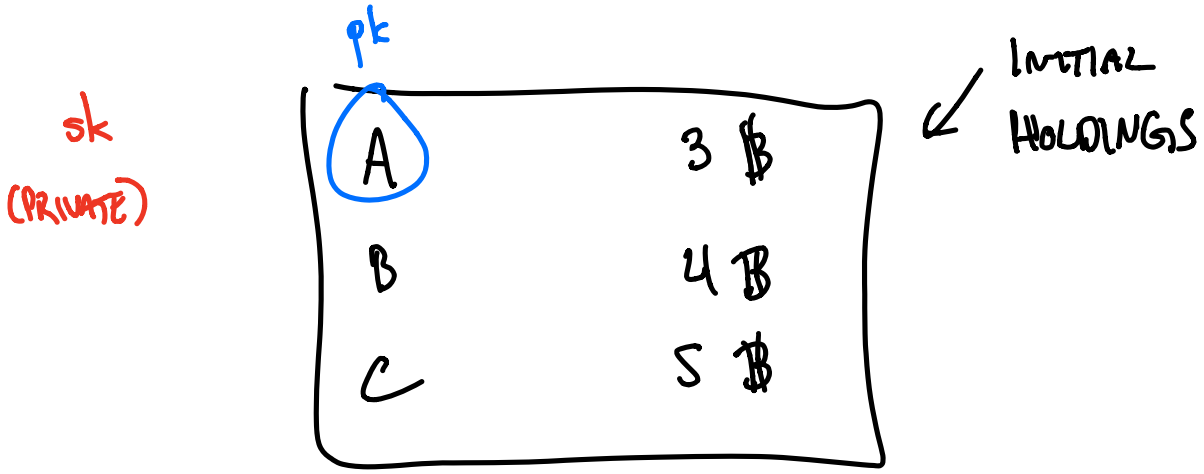
OUTPUT 0. THUS HONEST P_1 & P_2 OUTPUT

DIFFERENT VALUES. CONTRADICTION, THUS

NO SUCH PROTOCOL EXISTS. ☒

BLOCKCHAINS:

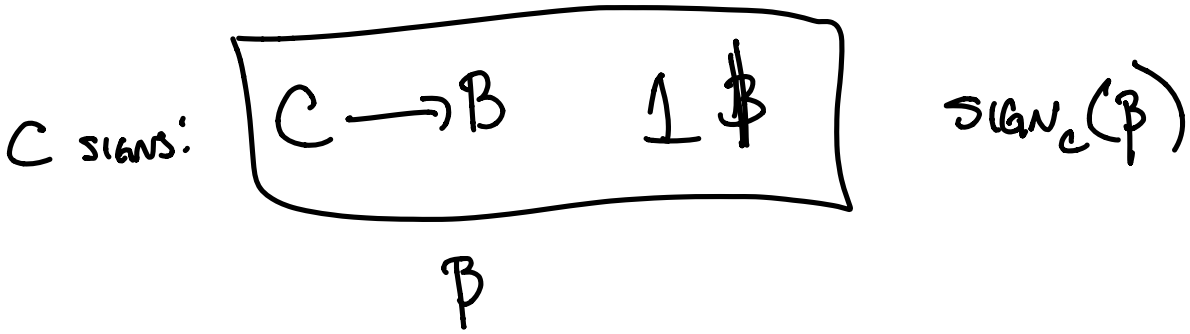
BLOCKCHAIN = APPEND-ONLY PUBLIC TRANSACTION LEDGER



EVERY IDENTITY IS pk FOR SIGNATURE SCHEME

CORRESPONDING sk IS "WALLET KEY"

HOW DO WE DO TRANSACTIONS?



MUST CHECK THAT TRANSACTION IS "VALID"

(C HAS AT LEAST 1 ₿)

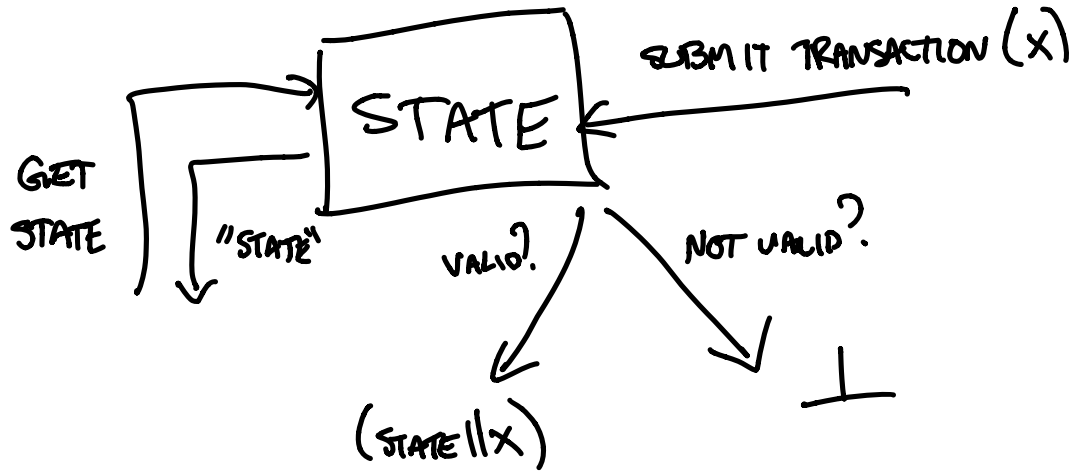
CAN DO THIS BASED ON INITIAL HOLDINGS

+ LIST OF TRANSACTIONS.

NOT QUITE WHAT ACTUALLY HAPPENS...

IN REALITY, WE USE A STATE MACHINE!

PUBLIC (APPEND-ONLY) TRANSACTION LEDGER :



STATE KEEPS TRACK OF BALANCES, CHECKS IF NEW TRANSACTIONS ARE VALID, & UPDATES BALANCES AFTER VALID TRANSACTIONS.

QUESTION: WHO MAINTAINS THE LEDGER?

DON'T WANT ONE SPECIFIC PARTY TO MAINTAIN: THEN THEY HAVE TOTAL CONTROL.

LET'S HAVE USERS COLLECTIVELY MAINTAIN LEDGER!

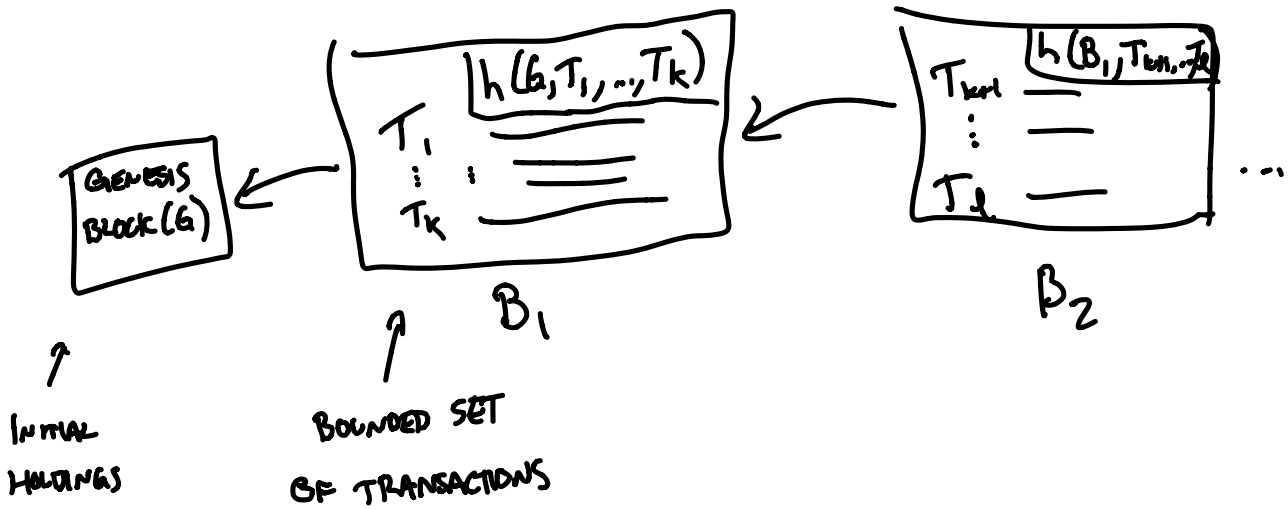
MUST MOTIVATE USERS TO DO THIS WORK: GIVE PORTION OF EACH TRANSACTION TO MAINTAINERS AS A REWARD!

So a BLOCKCHAIN is essentially:

A DECENTRALIZED, APPEND-ONLY PUBLIC TRANSACTION LEDGER.

Why is it called a BLOCKCHAIN?

THE LEDGER TAKES THE FOLLOWING FORM:



WHERE h IS CRHF. SO

IT'S A "CHAIN" OF BLOCKS CONTAINING

TRANSACTION INFO.

QUESTION: WHO DECIDES

WHICH BLOCKS GET ADDED

TO THE CHAIN?

Solution #1: "Permissioned" "Blockchain"

Choose a set of trusted servers

$$P_1, \ldots, P_n$$

who use MPC to determine if blocks are valid. If at least $\frac{n}{2}$ servers are not corrupt (or even $n-1$ depending on security model), then system is secure.

Solution #2: Permissionless Blockchain

Any user can act as a server & approve transactions! Can no longer assume honest majority of servers & use MPC, because Adversary could create arbitrary # of "users" & create dishonest majority.

Instead, we use various mechanisms (Proof of Work, Proof of Stake, etc.) for users to "prove" that they are a real user/computer & not one of millions of adversarily generated "users".

Proof of Work

Def (Puzzle-Friendly Hash Function)

For every output y it is

"somewhat" hard to find x

such that $h(k \| x) = y$

Example: Find x such that

$$H(B \| x) = \underbrace{000\ldots0}_{\text{20 0's}}\ldots$$

↑ Hardness parameter

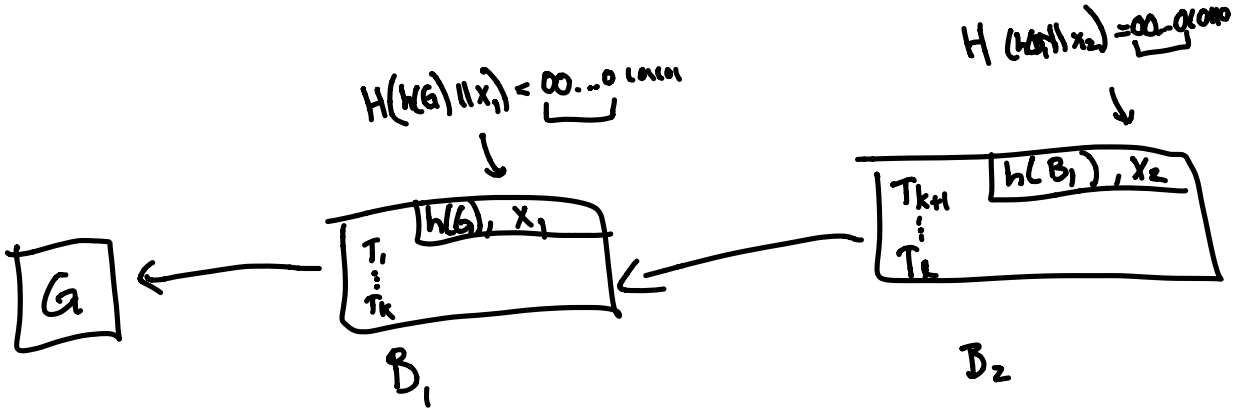
WHERE B IS SOME BLOCK OF TRANSACTIONS.

IF H HAS NO EXPLOITABLE STRUCTURE, NO BETTER STRATEGY THAN GUESSING RANDOM x OVER & OVER.

USER WHO FINDS SUCH Acc x "SOLVES" THE PUZZLE & USES THIS SOLUTION TO AUTHENTICATE BLOCK B (AND RECEIVE SOME REWARD).

THIS IS "MINING".

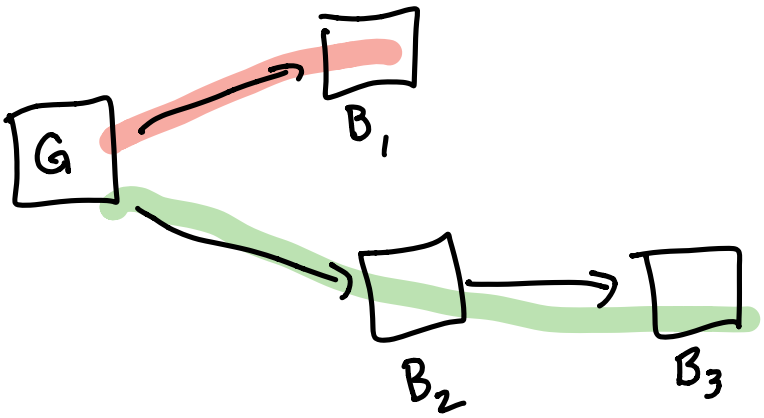
SO THE BLOCKCHAIN LOOKS LIKE THIS:



INCREASE # OF 0'S IN PREFIX (HARDNESS PARAMETER) WITH # OF PARTICIPANTS SO THAT, ON AVERAGE, IT TAKES ~10 MINUTES TO APPROVE A NEW BLOCK.

WHAT IF MULTIPLE SOLUTIONS ARE FOUND FOR DIFFERENT BLOCKS?

THIS IS CALLED A FORK.



HONEST MINERS ALWAYS CONTINUE WORKING FROM LONGEST CHAIN OF VALID TRANSACTIONS.

WHY IS THIS HELPFUL?

WE WANT TO MAKE SURE ADVERSARY CAN'T APPROVE INVALID TRANSACTIONS. WITH PROOF OF WORK, ADVERSARY WOULD NEED MORE COMPUTATIONAL POWER THAN ALL HONEST USERS APPROVING HONEST TRANSACTIONS. OTHERWISE, HONEST MINERS WILL CONTINUE TO APPROVE HONEST BLOCKS FASTER, & THE HONEST CHAIN WILL BE LONGER THAN THE ADVERSARIAL CHAIN.

WHY ~10 MINUTES TO APPROVE?

IF IT TAKES ~1 MINUTE TO PROPAGATE APPROVED BLOCK THROUGH NETWORK, PROBABILITY OF FORKING BECAUSE 2 SOLUTIONS ARE FOUND SIMULTANEOUSLY BECOMES SMALL WHEN TIME TO APPROVE BLOCKS IS LARGE.

FEWER FORKS MEANS A MORE CONSISTENT VIEW OF THE BLOCKCHAIN.

BUT IF TIME WAS TOO LARGE, WE WOULD HAVE TO WAIT A VERY LONG TIME TO BE SURE THAT A RECENT TRANSACTION IS REAL & NOT PART OF A FORK.

Byzantine Agreement vs. Proof of Work

In PoW, assuming players have equal compute power, players can agree if $t < \frac{n}{2}$

But in BA, we need $t < \frac{n}{3}$

Why this discrepancy?

In our BA proof, adversary had to simulate 4 players in this instead. So he does 2x as much work as honest players. Thus this proof doesn't apply when we consider adversary w/ less computing power than honest players.

PROOF OF STAKE:

USED BY ETHEREUM (2.0), CARDANO, ALGORAND

BUILDING BLOCK:

UNIQUE SIGNATURES: [GOLDWASSER, OSTROVSKY]

DIGITAL SIGNATURE SCHEME

KEY GEN
SIGN
VERIFY

$SIGN(D) = \sigma$

↑ ONLY POSSIBLE SIGNATURE FOR D

AT LEAST ONE BIT OF σ MUST BE UNPREDICTABLE, OTHERWISE σ CAN BE FORGED.

SIMILAR CONCEPT: VERIFIABLE RANDOM FUNCTION (VRF) [RABIN, KILIAN, MICALI]

UNIQUENESS & UNPREDICTABILITY HOLDS EVEN IF PUBLIC KEY IS CHOSEN MALICIOUSLY

HOW DOES THIS RELATE TO PROOF OF STAKE?

IDEA BEHIND PROOF OF STAKE (PoS):

(1) RUN A LOTTERY WHERE YOUR PROBABILITY OF WINNING DEPENDS ON HOW MUCH CURRENCY (STAKE) YOU HAVE IN THE CURRENCY / BLOCKCHAIN.

↖ DO THIS USING VRF / UNIQUE SIGNATURE

(2) IF YOU WIN LOTTERY, ANNOUNCE IT

(3) USE BYZANTINE AGREEMENT AMONG SMALL # OF WINNERS TO AGREE ON NEXT BLOCK

How do we run the lottery?

Players sign previous block w/ their secret key. Win if signature has certain # of zeroes as prefix (# of zeroes necessary depends on how much stake they have). Can prove you won by publishing your signature for others to verify.

Security Guarantee:

If $< \frac{1}{3}$ of total currency is held by adversary, $< \frac{1}{3}$ of Byzantine agreement players are dishonest with high probability.

And they approve an honest block. Otherwise, no progress is made until next lottery.

IF $> \frac{2}{3}$ ARE DISHONEST, A DISHONEST COMMITTEE CAN CREATE A FORK. LONGEST CHAIN RULE DOESN'T QUITE WORK, NEED MORE ADVANCED RULE (NOT DISCUSSED HERE).

CARDANO APPROACH: ONLY 1 WINNER OF LOTTERY, NO BYZANTINE AGREEMENT. "TRUST THE KING"

SMART CONTRACTS :

STANDARD CRYPTOCURRENCIES / BLOCKCHAINS

CONTAIN ONLY TRANSACTIONS ON THE LEDGER.

WHAT IF WE ALLOW CONDITIONAL STATEMENTS BASED ON ARBITRARY PROGRAMS?

IF $P($100 FUTURE BLOCKS$) =$ TRUE :

THEN A $\longrightarrow$ B 1 ETH

EXAMPLE: IF YOU PUBLISH LEGAL CERTIFICATE THAT YOU TRANSFERRED CAR TITLE TO ME, THEN I WILL TRANSFER TO YOU 5 ETH

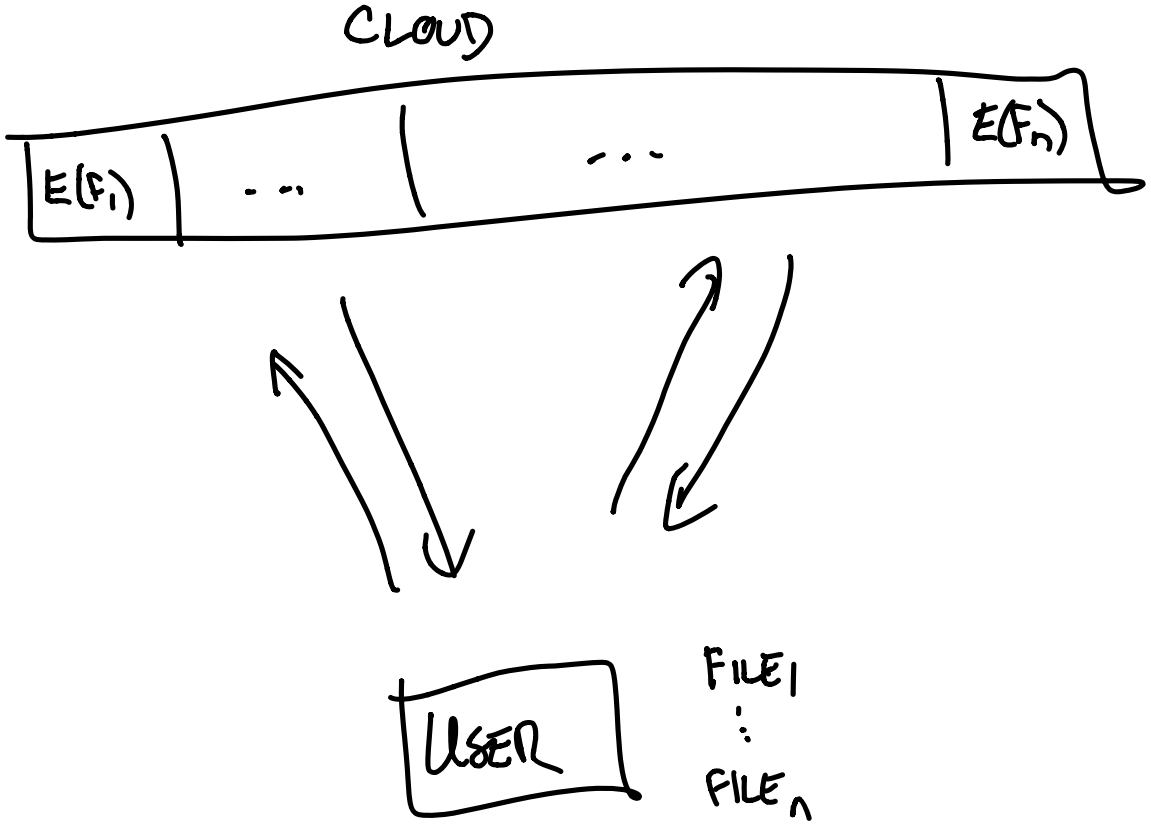
IN PRACTICE, P IS NOT ARBITRARY CODE, BECAUSE YOU MUST PAY "GAS" TO MINERS BASED ON HOW COMPLEX THE PROGRAM IS.

SO SHORTER PROGRAMS ARE CHEAPER.

Sneak Peek at Next Week:

Oblivious RAM!

Suppose you're using Dropbox for files



PROBLEM: CLOUD SEES ACCESS PATTERN OF FILES.

E.G. "YOU ACCESSED THE SAME FILE 3 TIMES TODAY"

IF YOU KNOW ARNOLD SCHWARZENEGGER WAS HOSPITALIZED ON TWO DIFFERENT DATES, CAN USE INTERSECTION OF ACCESS PATTERN OF HOSPITAL'S PRIVATE RECORDS TO LEARN WHICH RECORD BELONGS TO ARNOLD.

THEN, CAN KNOW WHEN HE IS HOSPITALIZED AGAIN WHENEVER THIS RECORD IS ACCESSED.

ORAM HIDES ACCESS PATTERN FOR THESE ENCRYPTED FILES.