Coinshuffle++

P2P MIXING AND UNLINKABLE BITCOIN TRANSACTIONS

Ruffing, Moreno-Sanchez and Kate (2017)

P2P Mixing and Unlinkable Bitcoin Transactions

Anonymity of the people, by the people, and for the people

Tim Ruffing
CISPA, Saarland University
tim.ruffing@mmci.uni-saarland.de

Pedro Moreno-Sanchez
Purdue University
pmorenos@purdue.edu

Aniket Kate
Purdue University
aniket@purdue.edu

***Abstract*—Starting with Dining Cryptographers networks (DC-net), several peer-to-peer (P2P) anonymous communication protocols have been proposed. Despite their strong anonymity guarantees none of those has been employed in practice so far: Most fail to simultaneously handle the crucial problems of slot collisions and malicious peers, while the remaining ones handle those with a significant increased latency (communication rounds) linear in the number of participating peers in the best case, and *quadratic* in the worst case. We conceptualize these P2P anonymous communication protocols as *P2P mixing*, and present a novel P2P mixing protocol, DiceMix, that only requires constant (i.e., four) communication rounds in the best case, and $4 + 2f$ rounds in the worst case of f malicious peers. As every individual malicious peer can prevent a protocol run from success by omitting his messages, we find DiceMix with its worst-case linear-round complexity to be an optimal P2P mixing solution.**

Starting with the dining cryptographers network (DC-net) protocol [14], another line of ACNs research emerged, where users (or peers) do *not* employ any third party proxies and instead communicate with each other to send their messages anonymously. While the DC-net protocol can guarantee successful termination and anonymity against honest-but-curious adversary controlling a subset of users (or peers), it is easily prone to disruption by a single malicious peer who sends invalid protocol messages (active disruption) or omits protocol messages entirely (crash). Moreover, a DC-net protects the anonymity of the involved malicious peers and subsequently cannot ensure termination of the protocol run by detecting and excluding the malicious peer.

To address this termination issue, several recent successors

Two Weeks ago - CoinShuffle

- One issue with CoinJoin implementations is the reliance on a central coordinator**. Removing the coordinator** would require a secure method of participants declaring their anonymous addresses
- We can replace the coordinator with a Coin*Shuffle,* where each participant **onion-encrypts their address** with the public keys of the latter participants. They then **decrypt and shuffle** all encrypted addresses they have received with their own address, and proceed to hand off the encrypted addresses to the next participant.
- **Scales poorly** with many participants, ElectronCash(5)

Wasabi Research Club

- January 6th, 2020 – Knapsack CoinJoin
- January 13th, 2020 – SNICKER
- January 20th, 2020 – CoinShuffle
- January 27th, 2020 – Dining Cryptographer Networks
- February 3rd, 2020 – CoinShuffle ++
- Feburary 10th, 2020 - TBD

https://github.com/zkSNACKs/WasabiResearchClub

Wasabi Research Club

- January 6th, 2020 – Knapsack CoinJoin
- January 13th, 2020 – SNICKER
- January 20th, 2020 – CoinShuffle
- January 27th, 2020 – Dining Cryptographer Networks
- February 3rd, 2020 – CoinShuffle ++
- Feburary 10th, 2020 - TBD

https://github.com/zkSNACKs/WasabiResearchClub

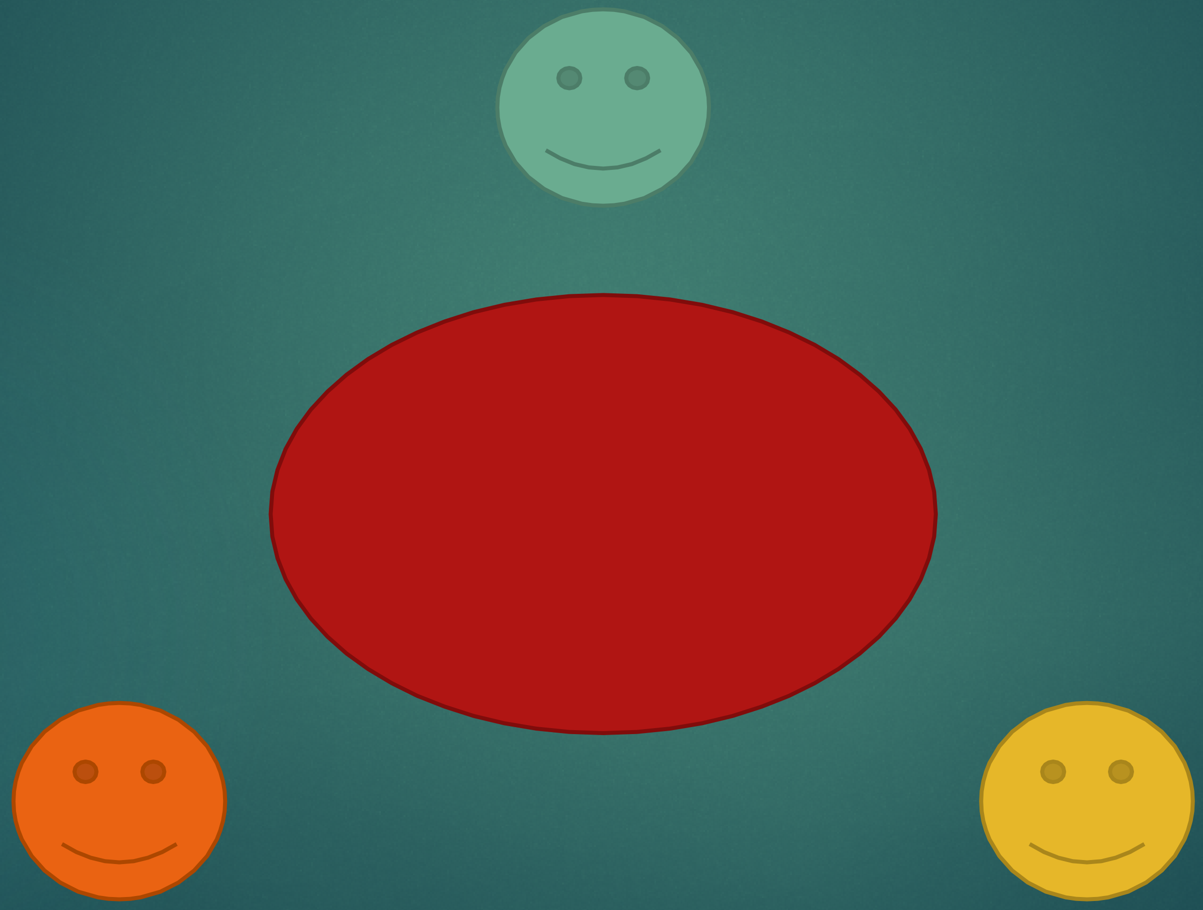
Dining Cryptographers Recap

- *Three cryptographers are sitting down to dinner at their favorite three-star restaurant. Their waiter informs them that arrangements have been made with the maitre d'hotel for **<u>the bill to be paid anonymously. One of the cryptographers might be paying for the dinner, or it might have been NSA</u>** (U.S. National Security Agency). The three cryptographers respect each other's right to make an anonymous payment, but they wonder if NSA is paying. They resolve their uncertainty fairly by carrying out the following protocol- (Chaum, 1988)*

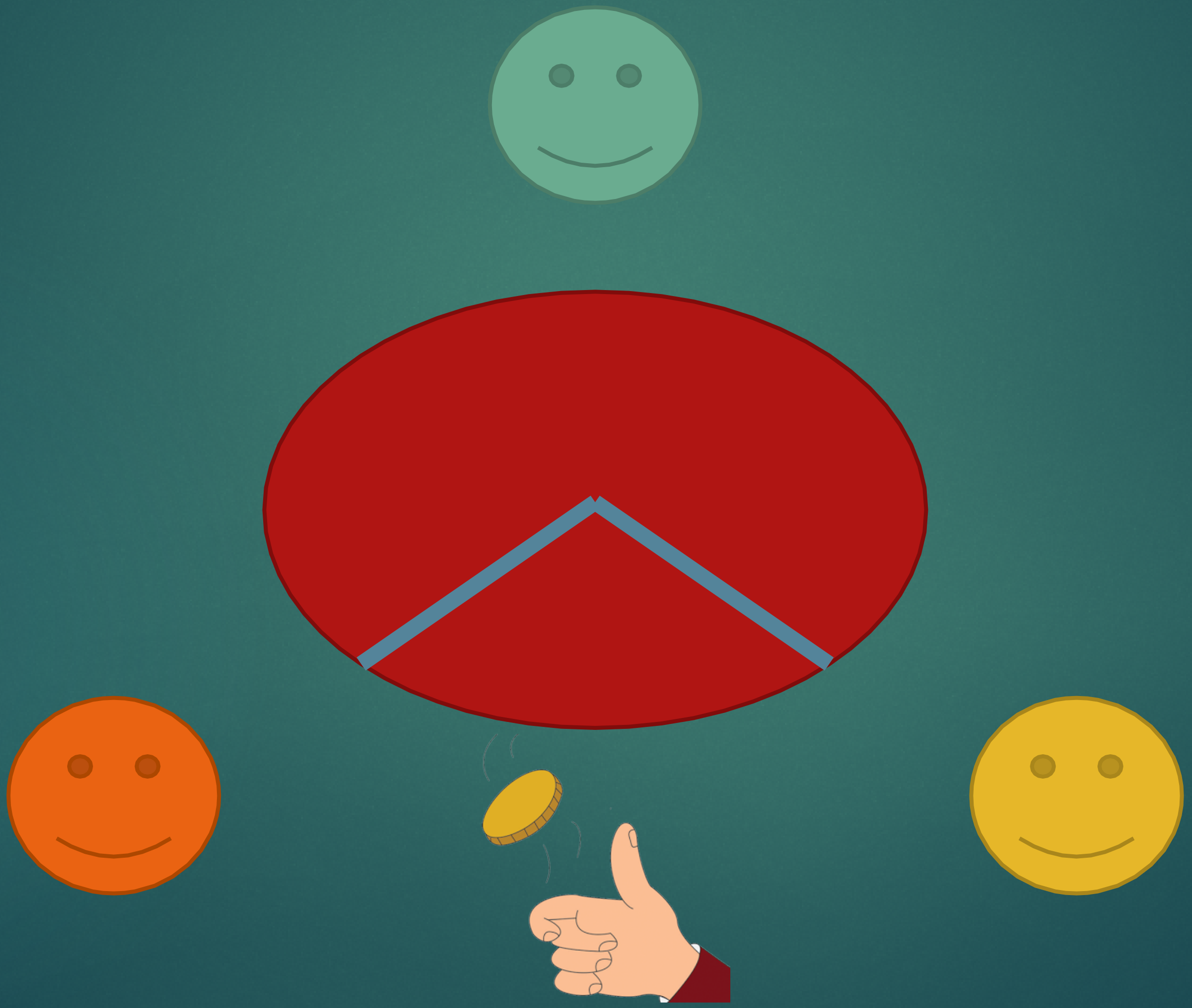
Dining Cryptographers Recap

- *Anonymous communication among **<u>honest</u>** peers.*
- *Protocol is **<u>resistant to traffic analysis</u>**, in contrast to something like Tor which is not.*

The Premise – The Cryptographers at Dinner



The Premise – The Cryptographers at Dinner

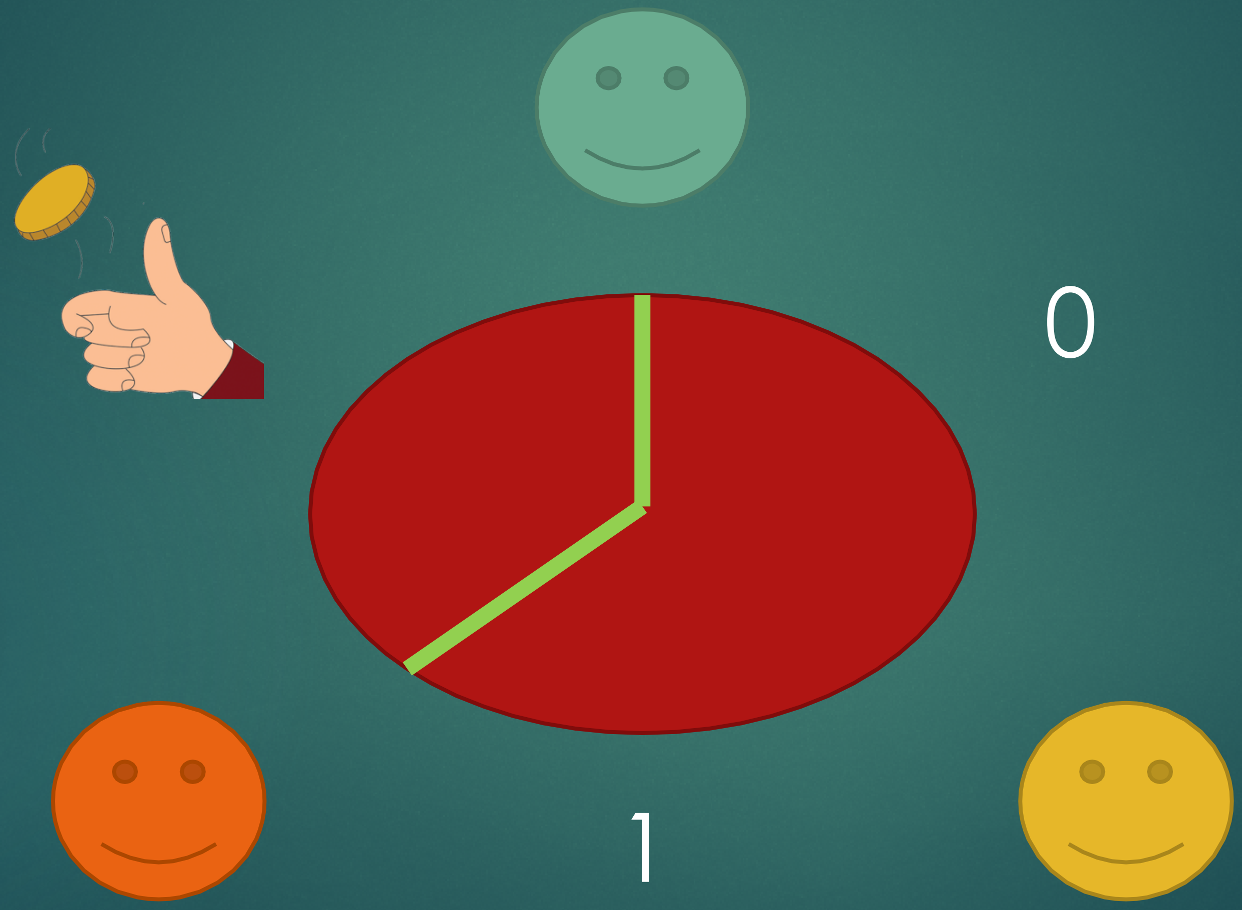


The Premise – The Cryptographers at Dinner

The Premise – The Cryptographers at Dinner

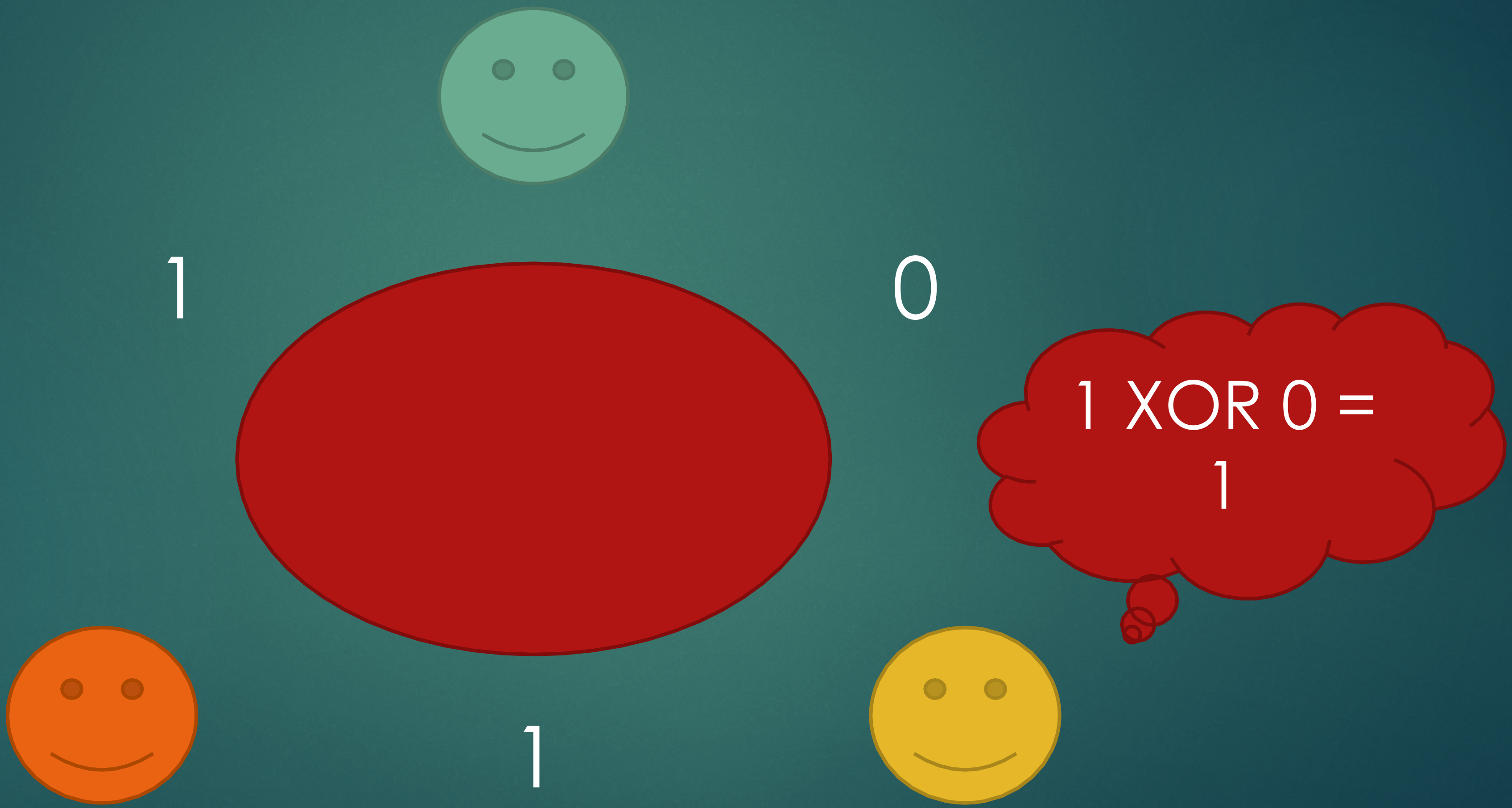
The Premise – The Cryptographers at Dinner

The Premise – The Cryptographers at Dinner

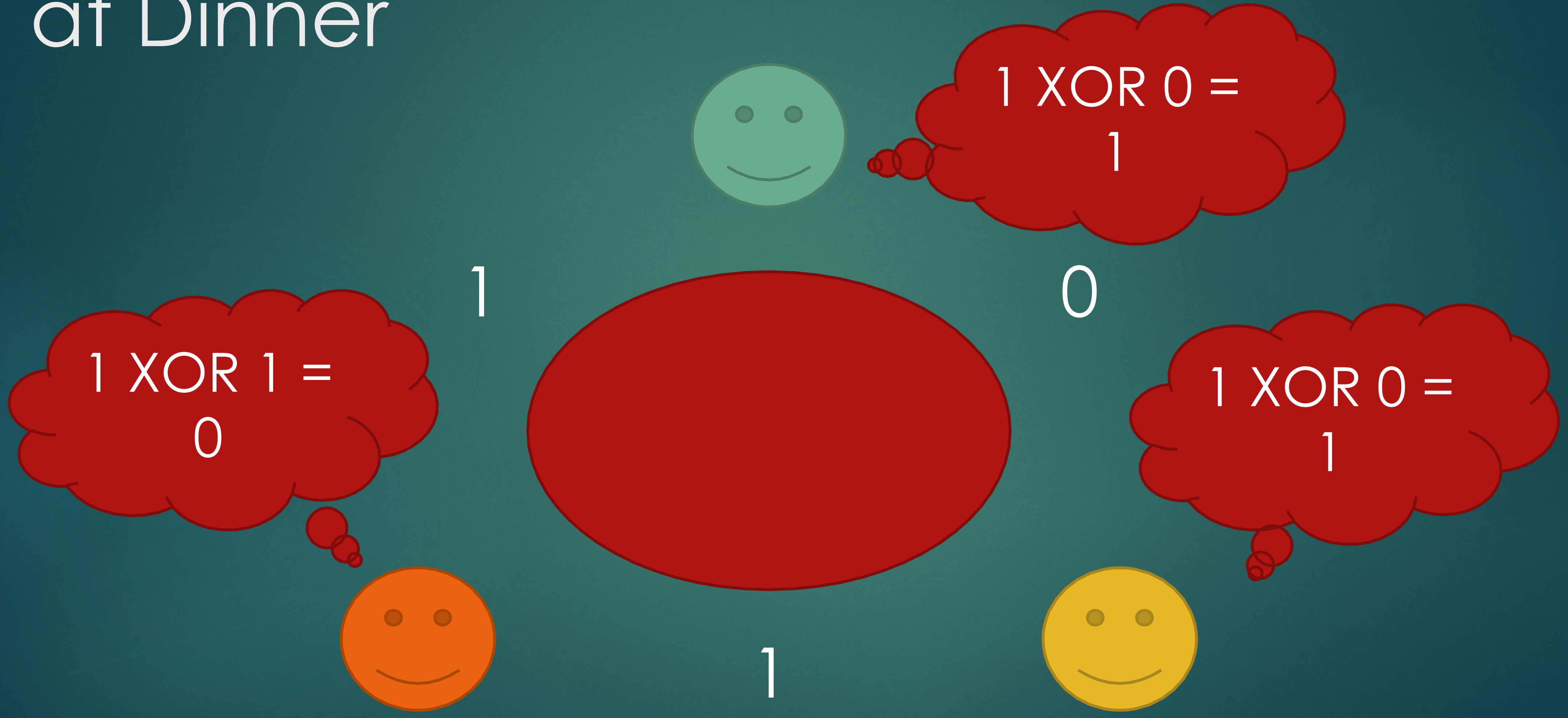


The Premise – The Cryptographers at Dinner

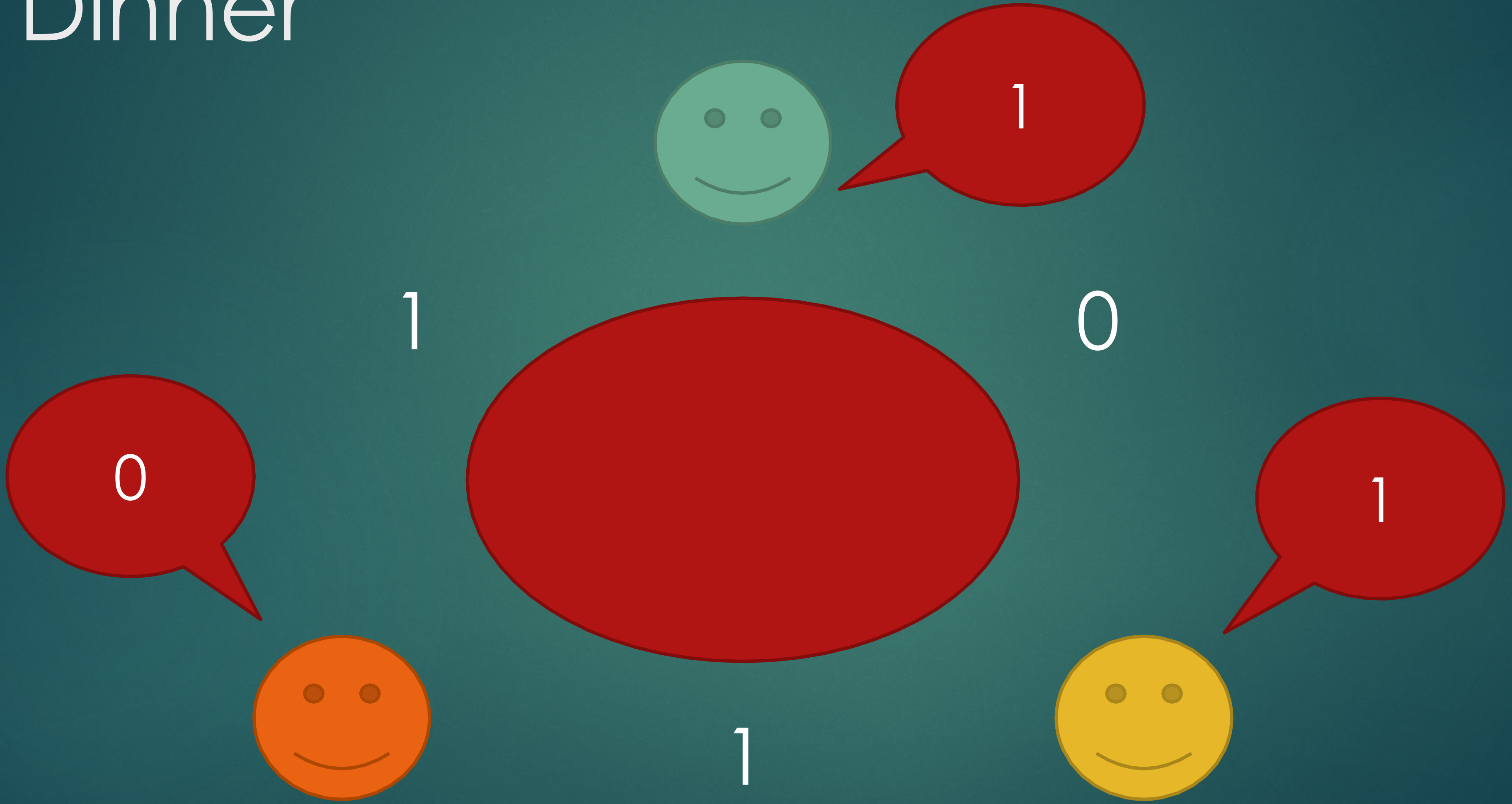
The Premise – The Cryptographers at Dinner



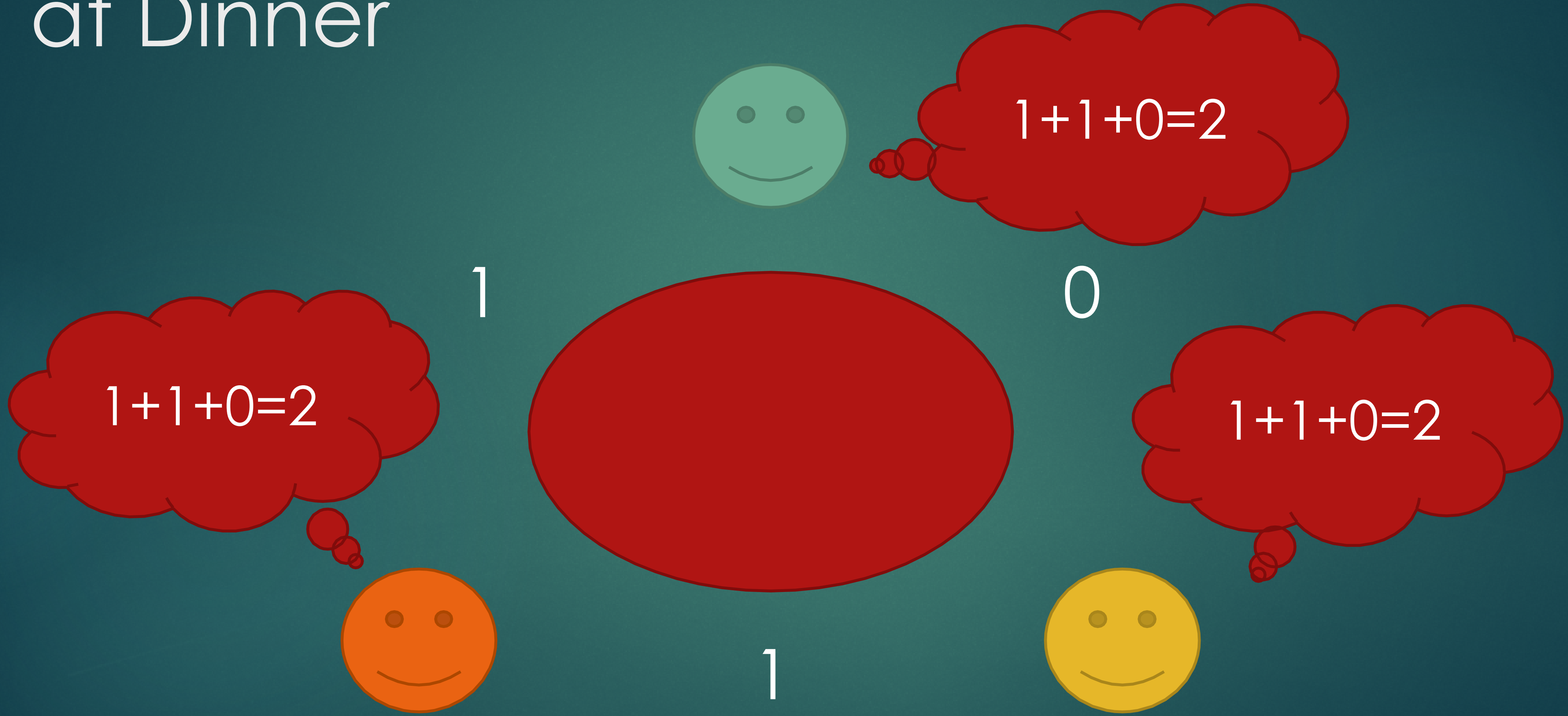
The Premise – The Cryptographers at Dinner



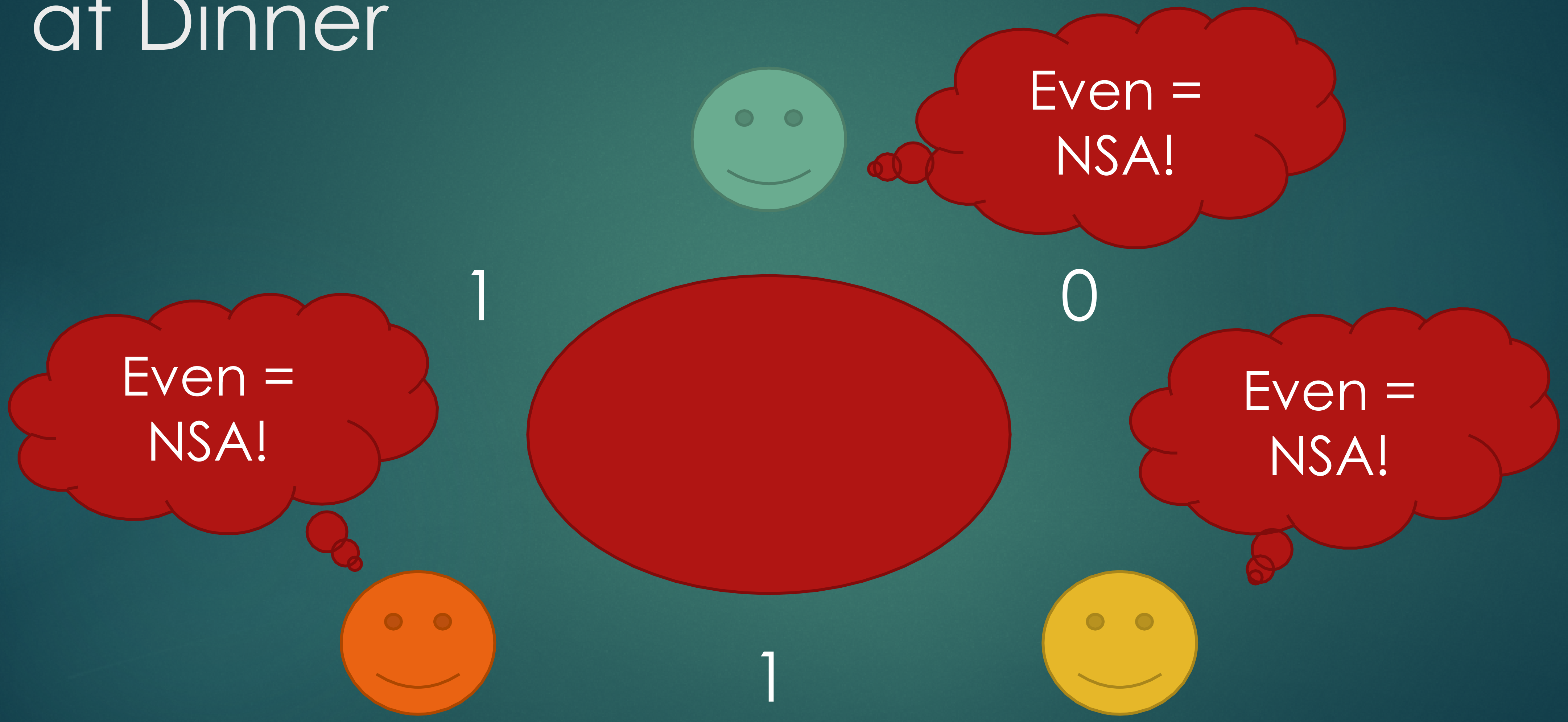
The Premise – The Cryptographers at Dinner



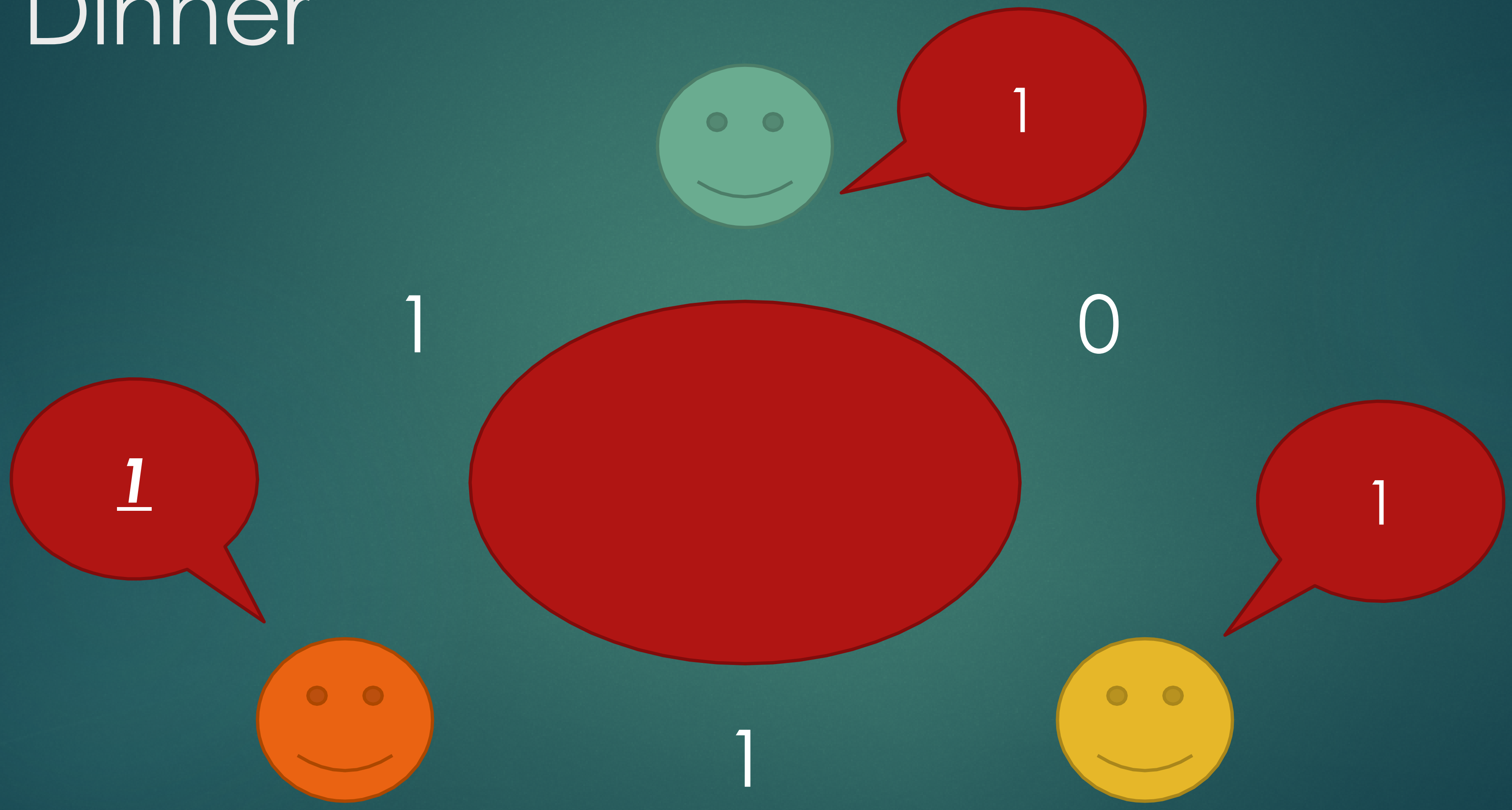
The Premise – The Cryptographers at Dinner



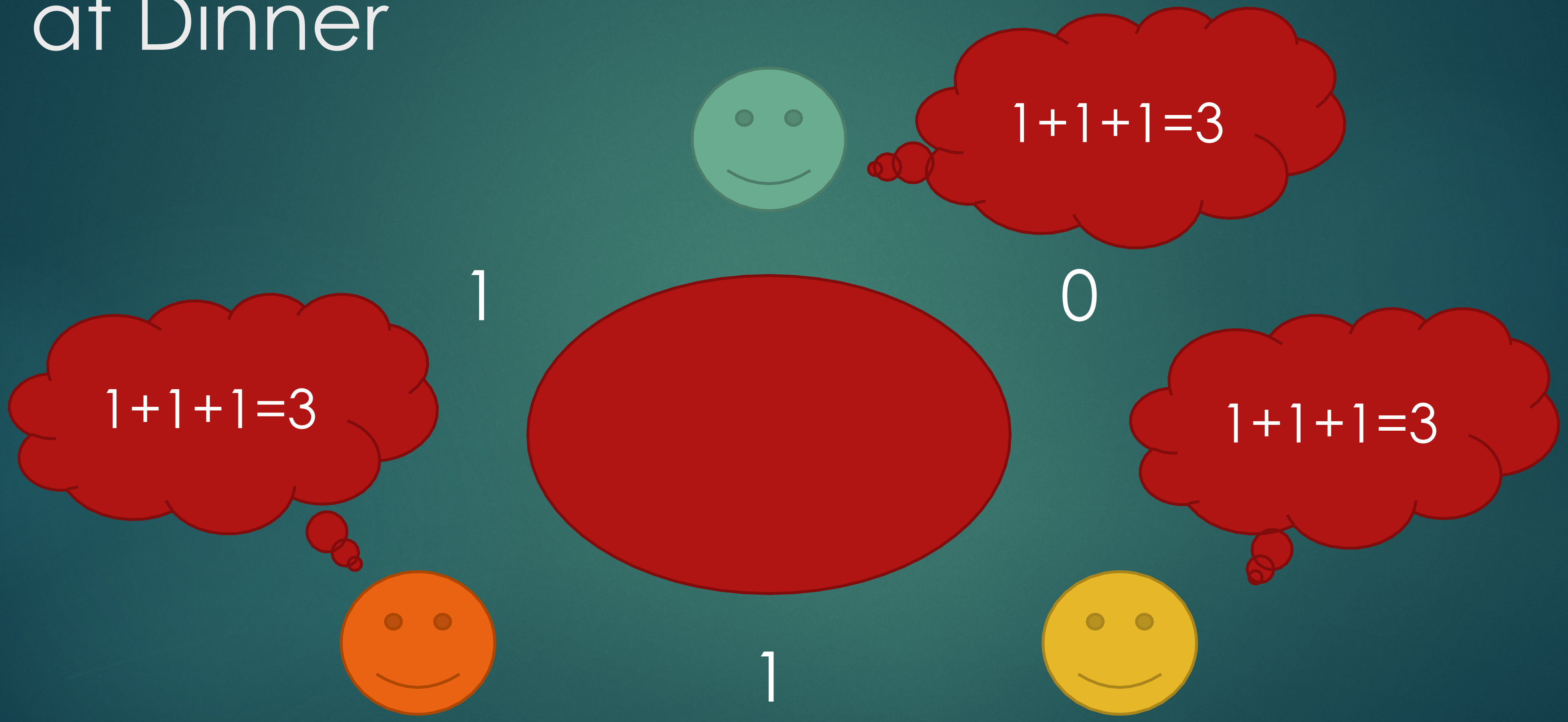
The Premise – The Cryptographers at Dinner



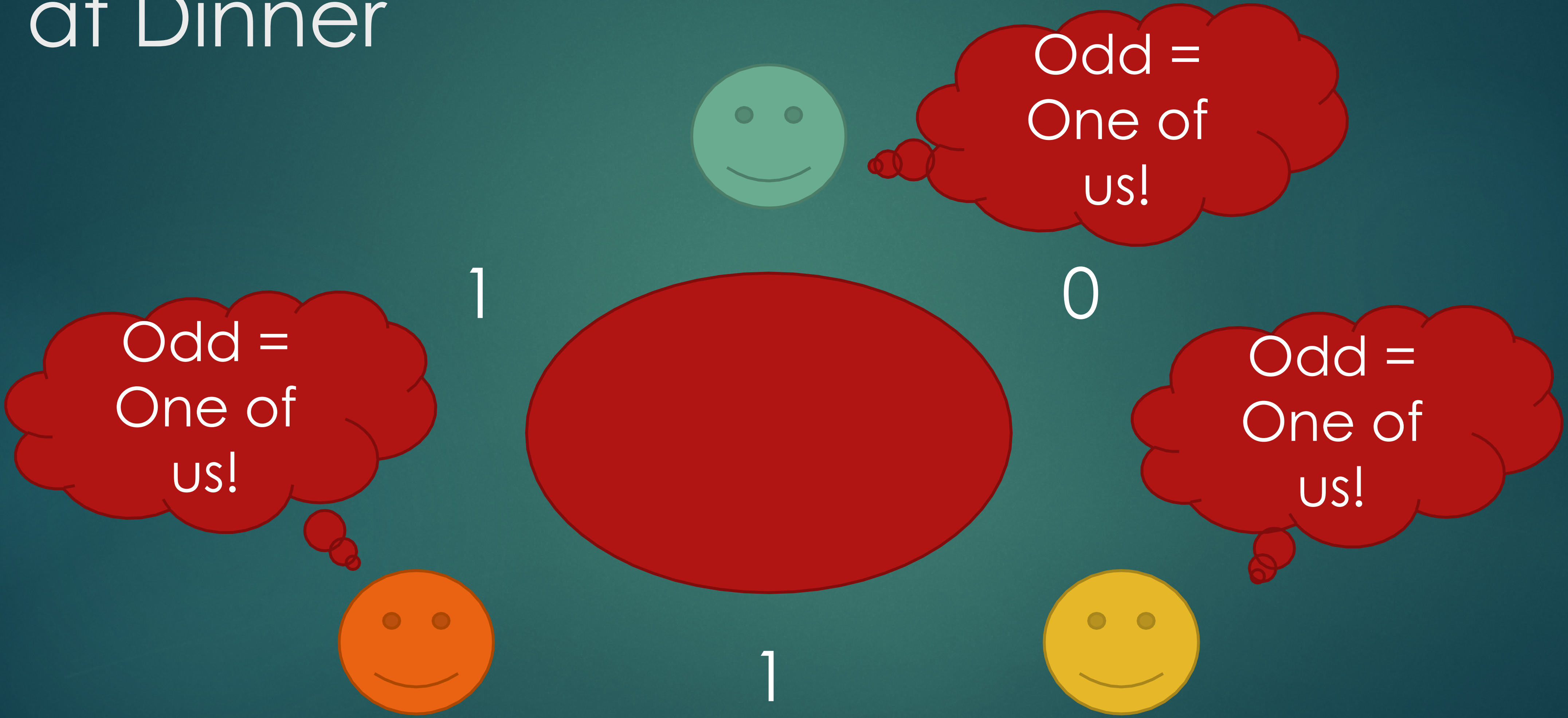
The Premise – The Cryptographers at Dinner



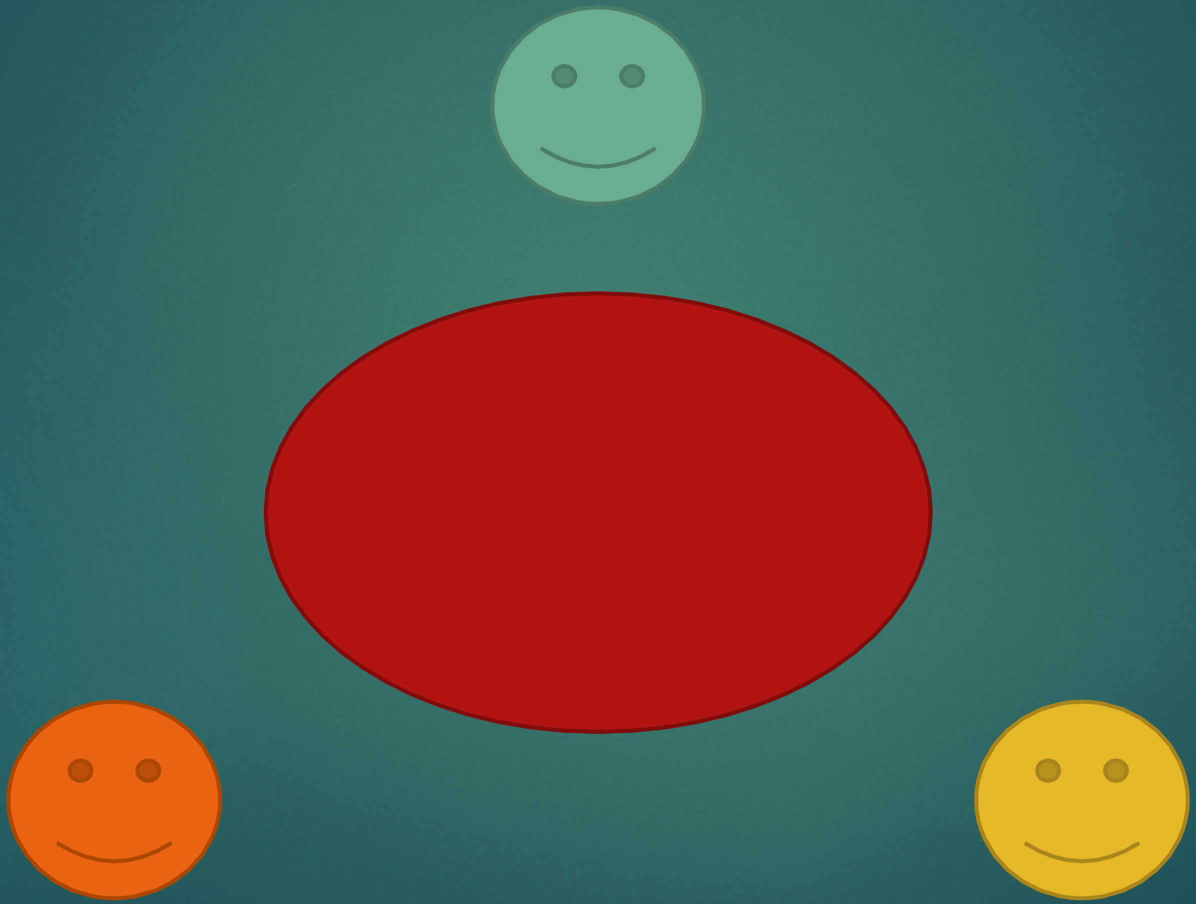
The Premise – The Cryptographers at Dinner



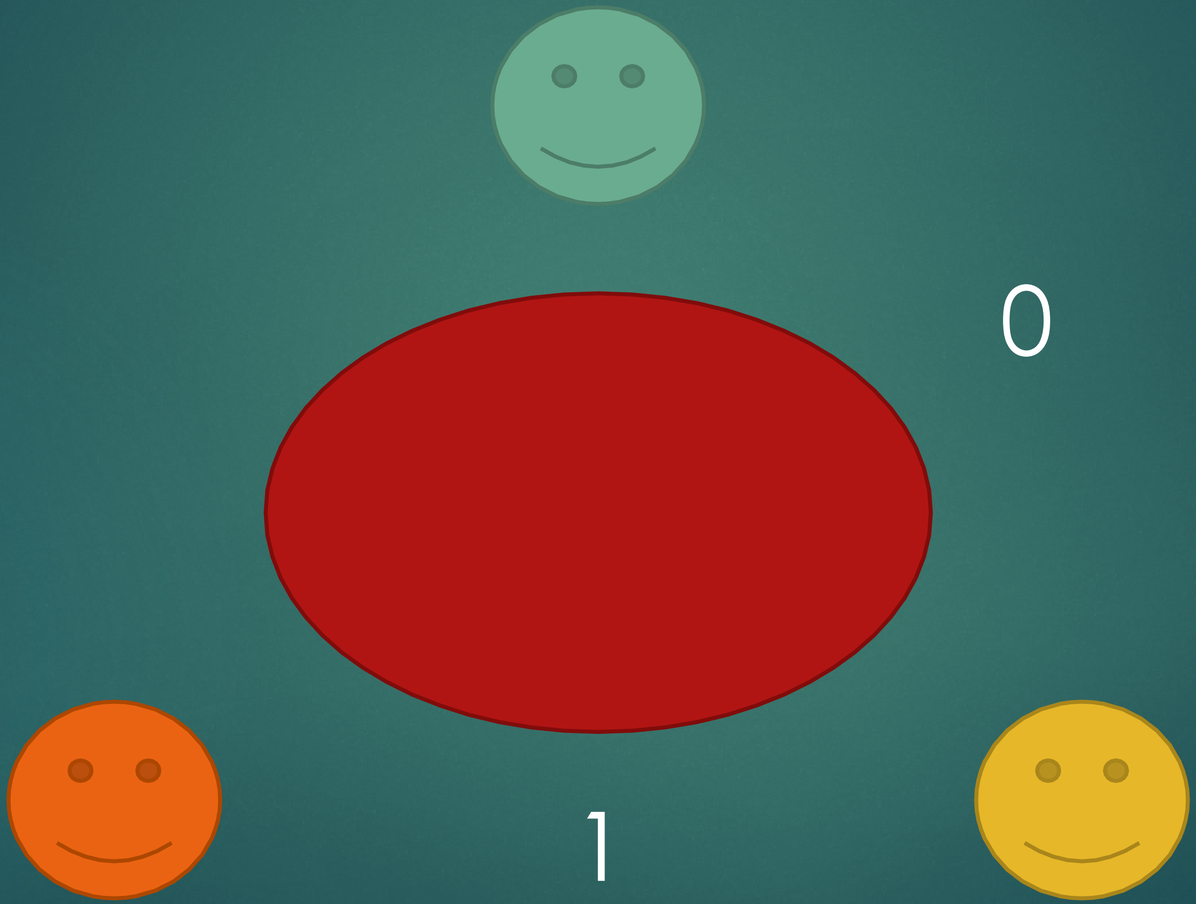
The Premise – The Cryptographers at Dinner



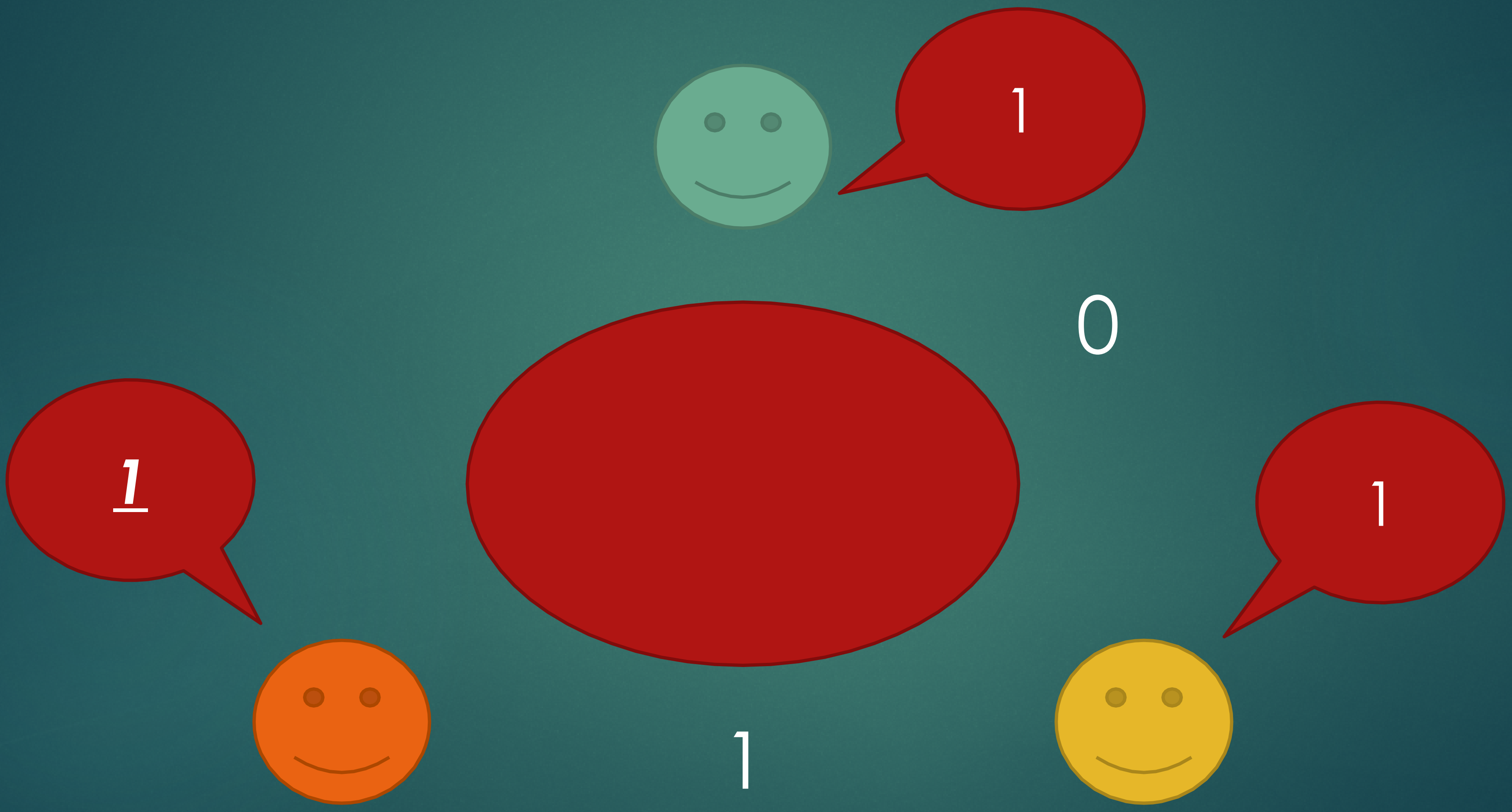
Why does this work?



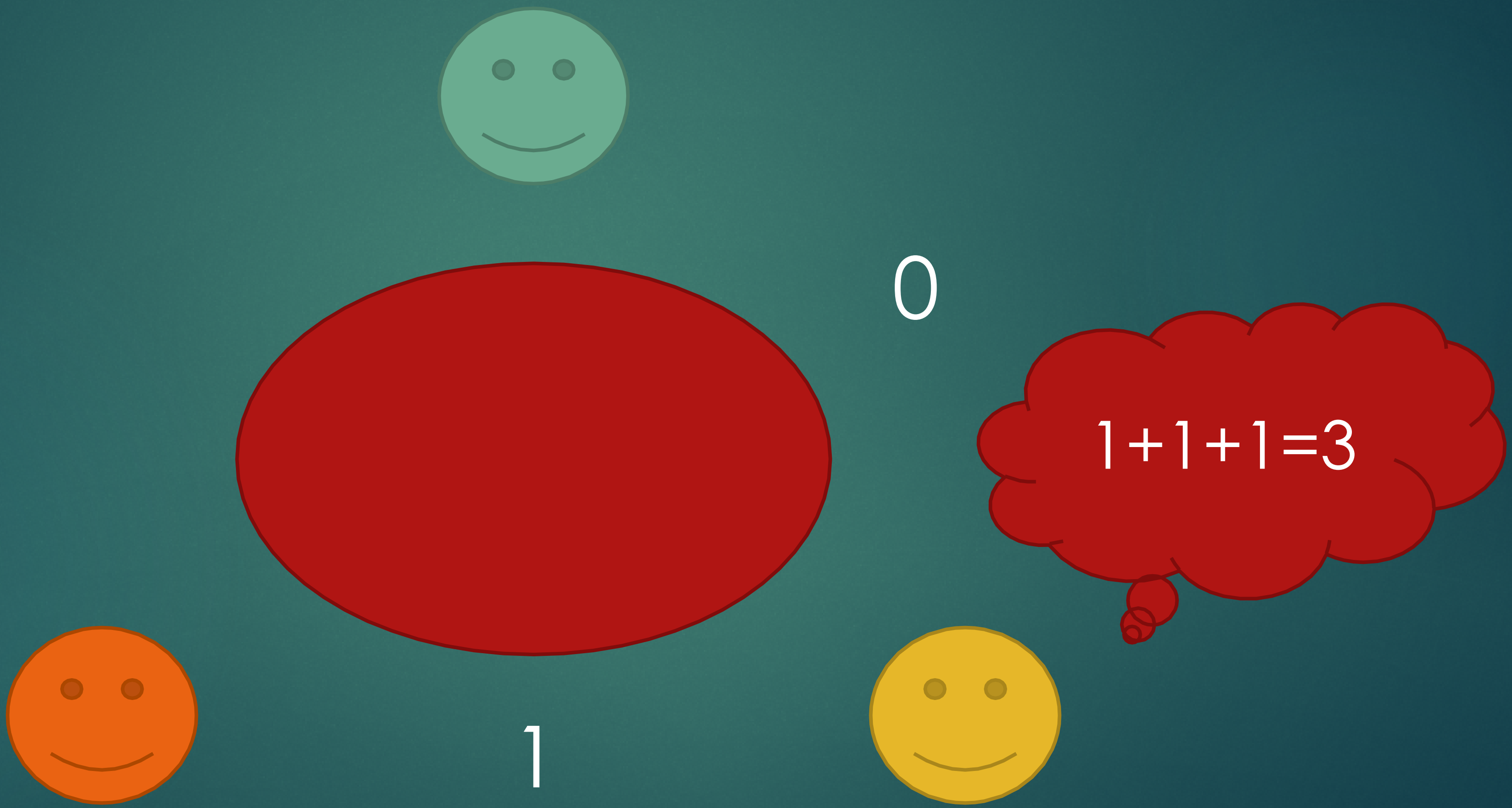
Why does this work?



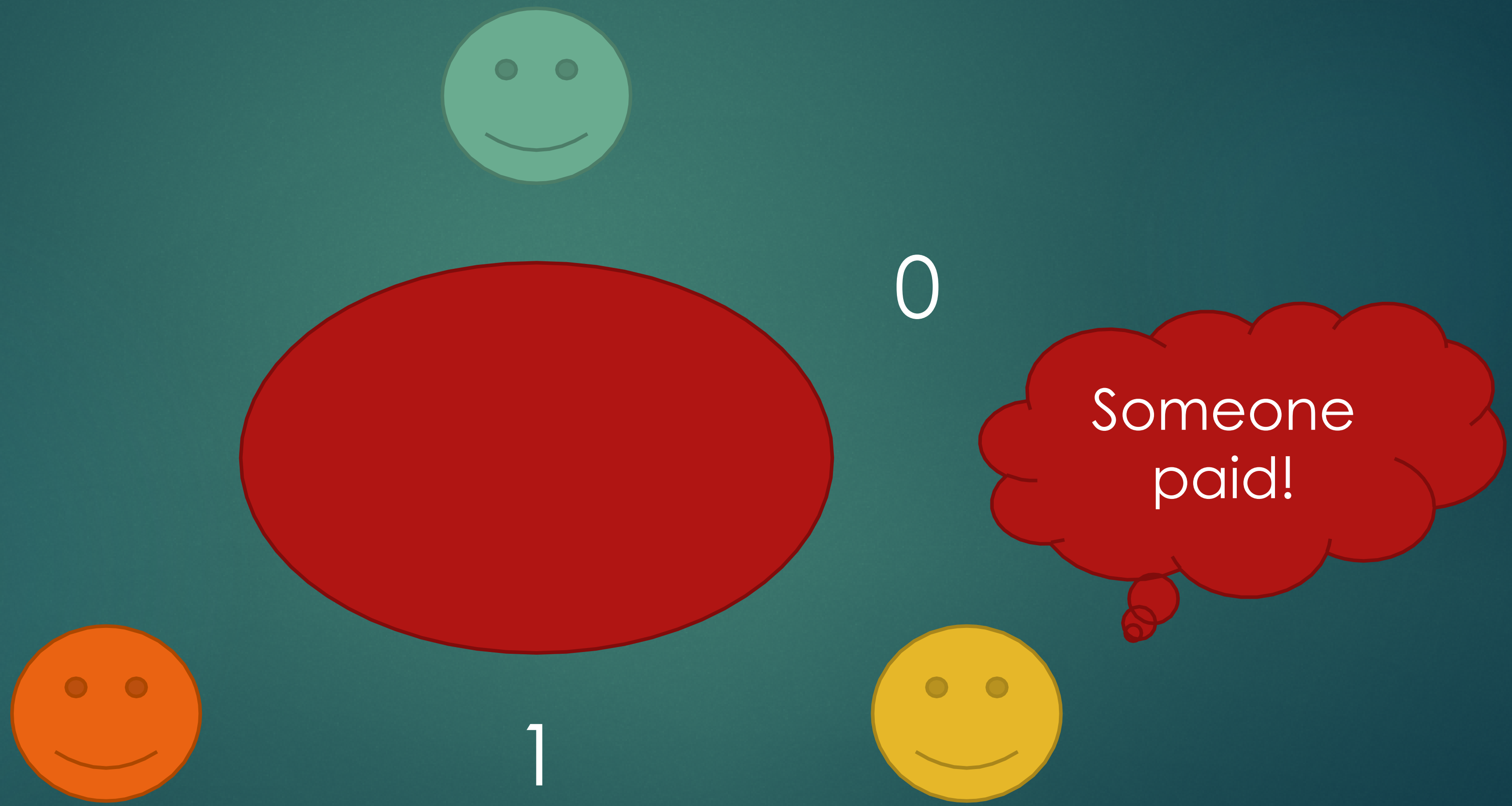
Why does this work?



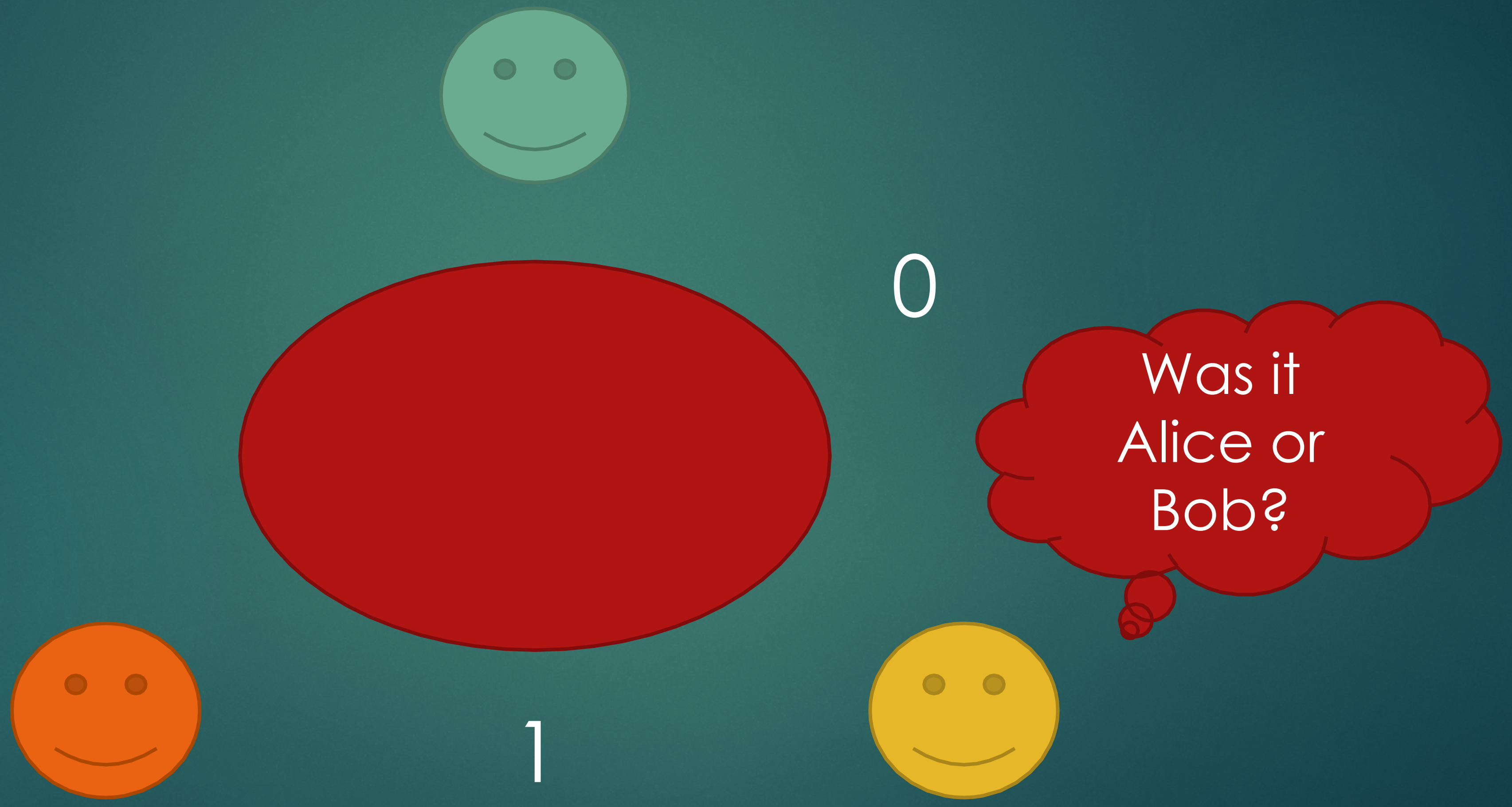
Why does this work?



Why does this work?

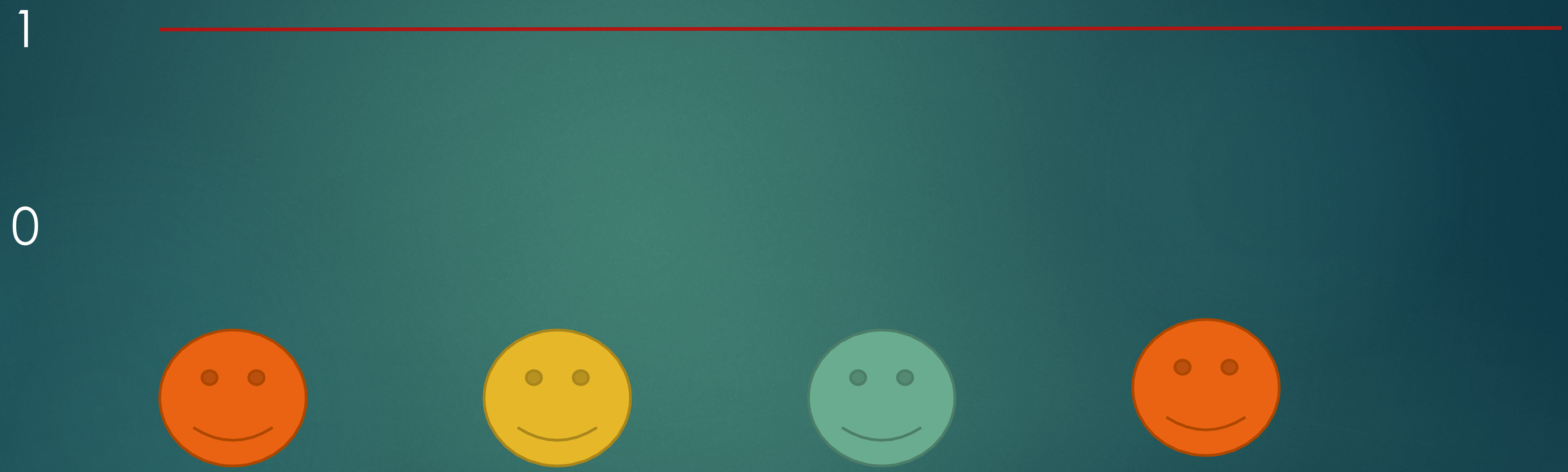


Why does this work?



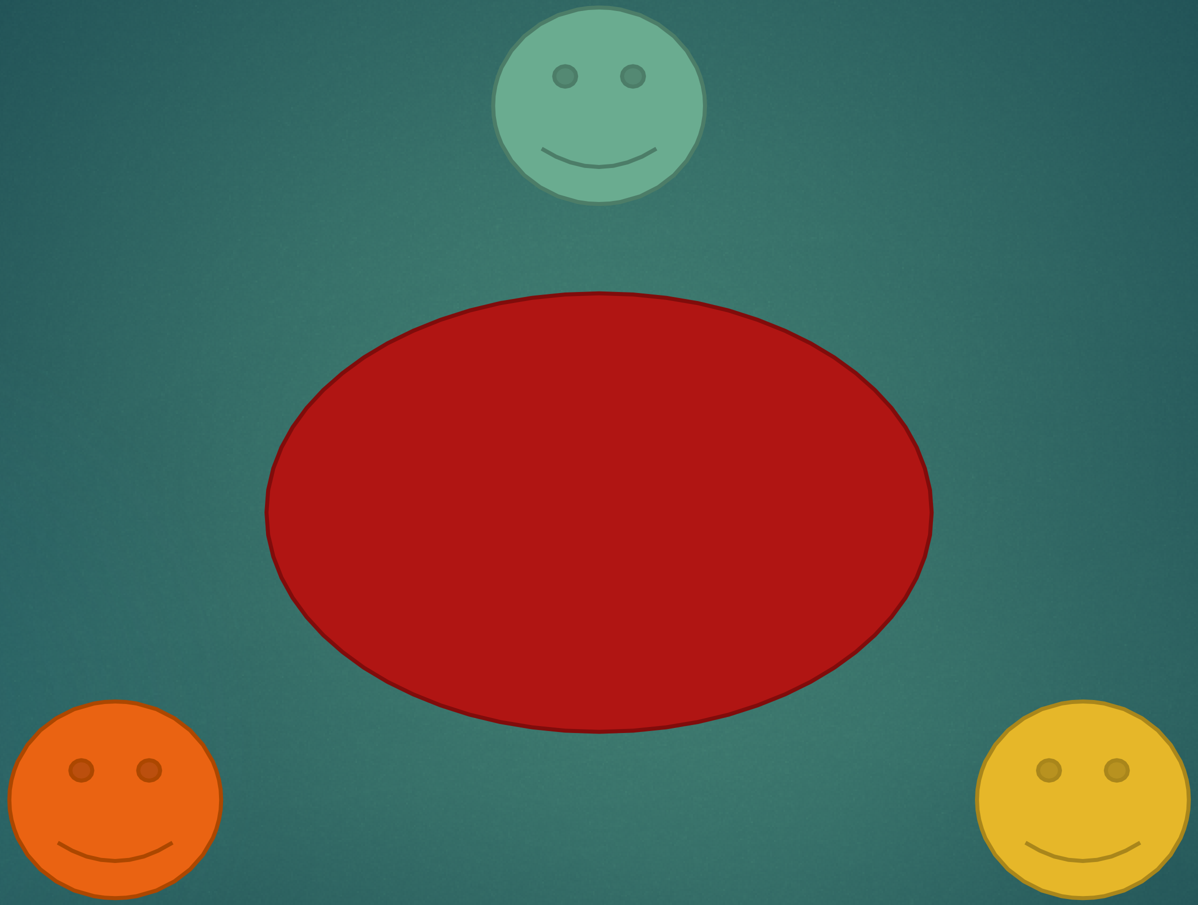
Why is it always even?

Why is it always even?

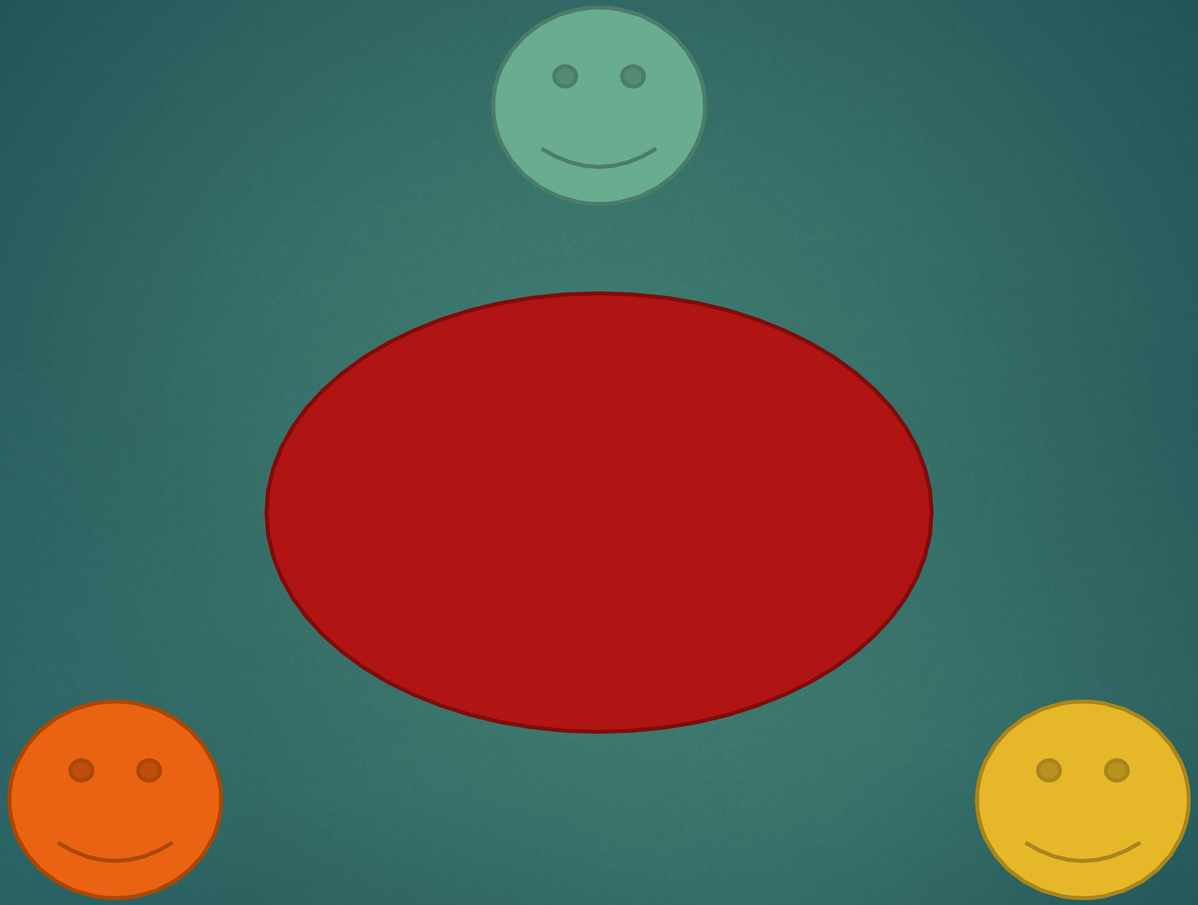


Why is it always even?

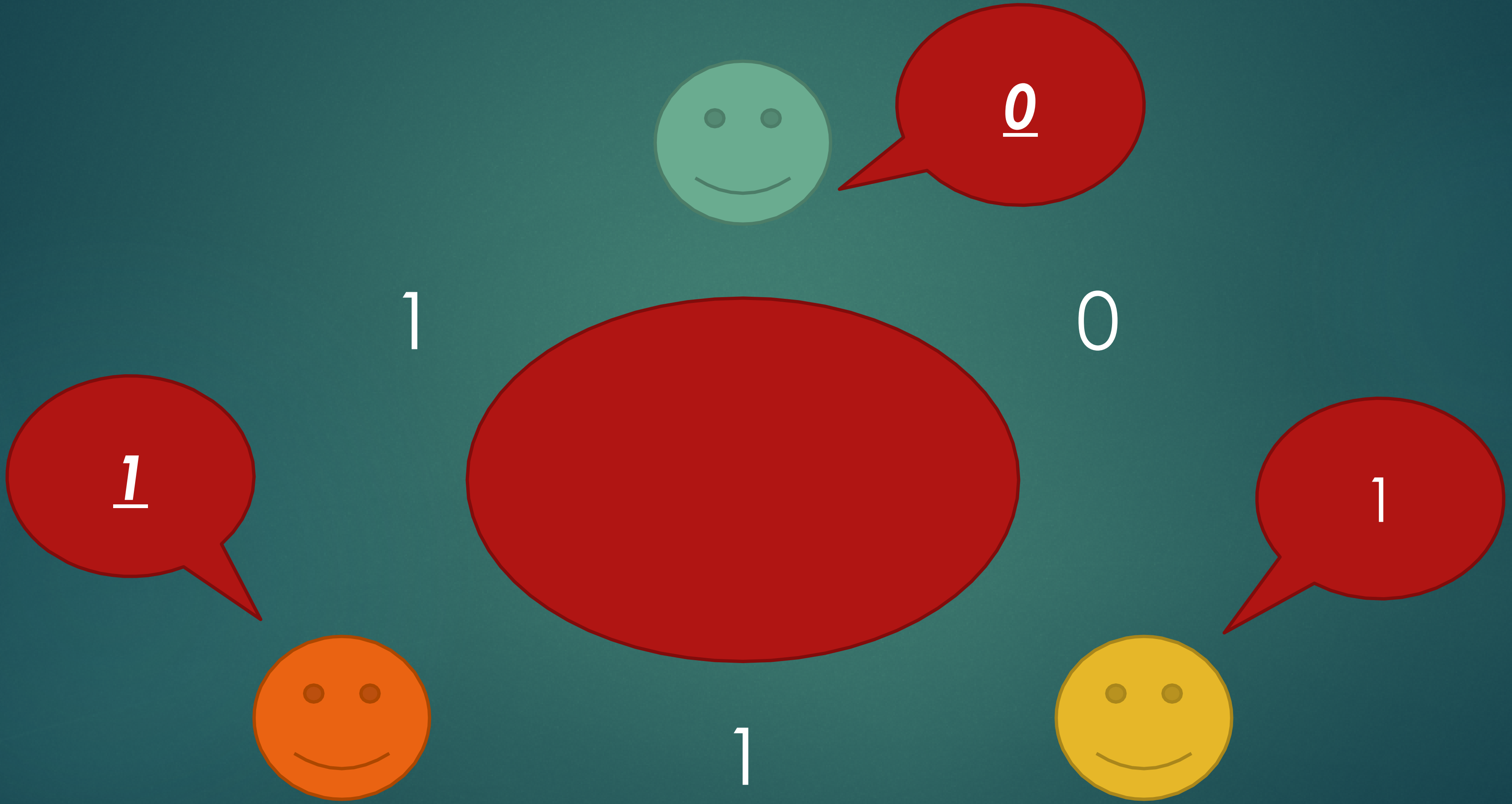
Problems with DC Nets



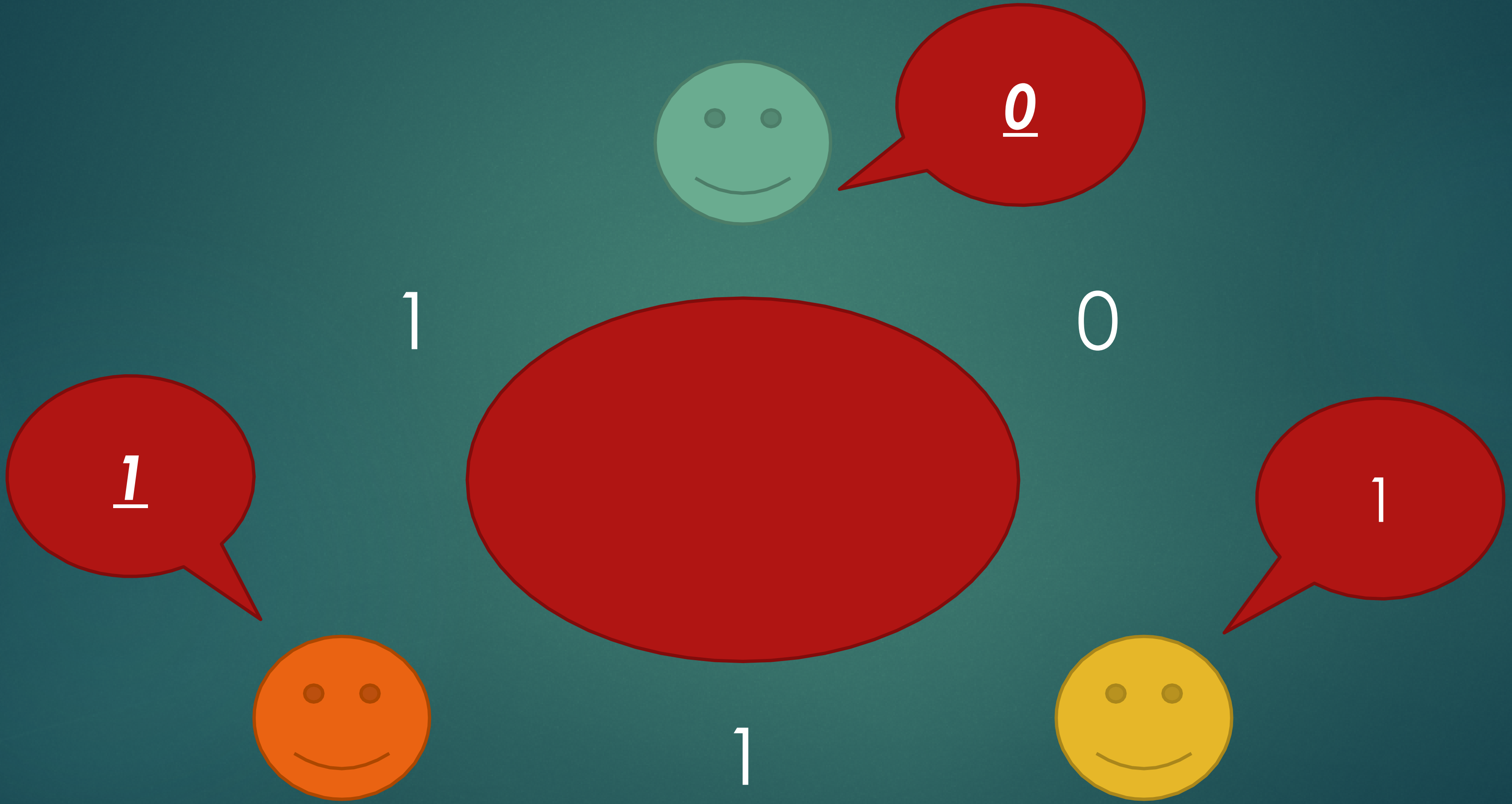
Problem 1 – Collision of Messages



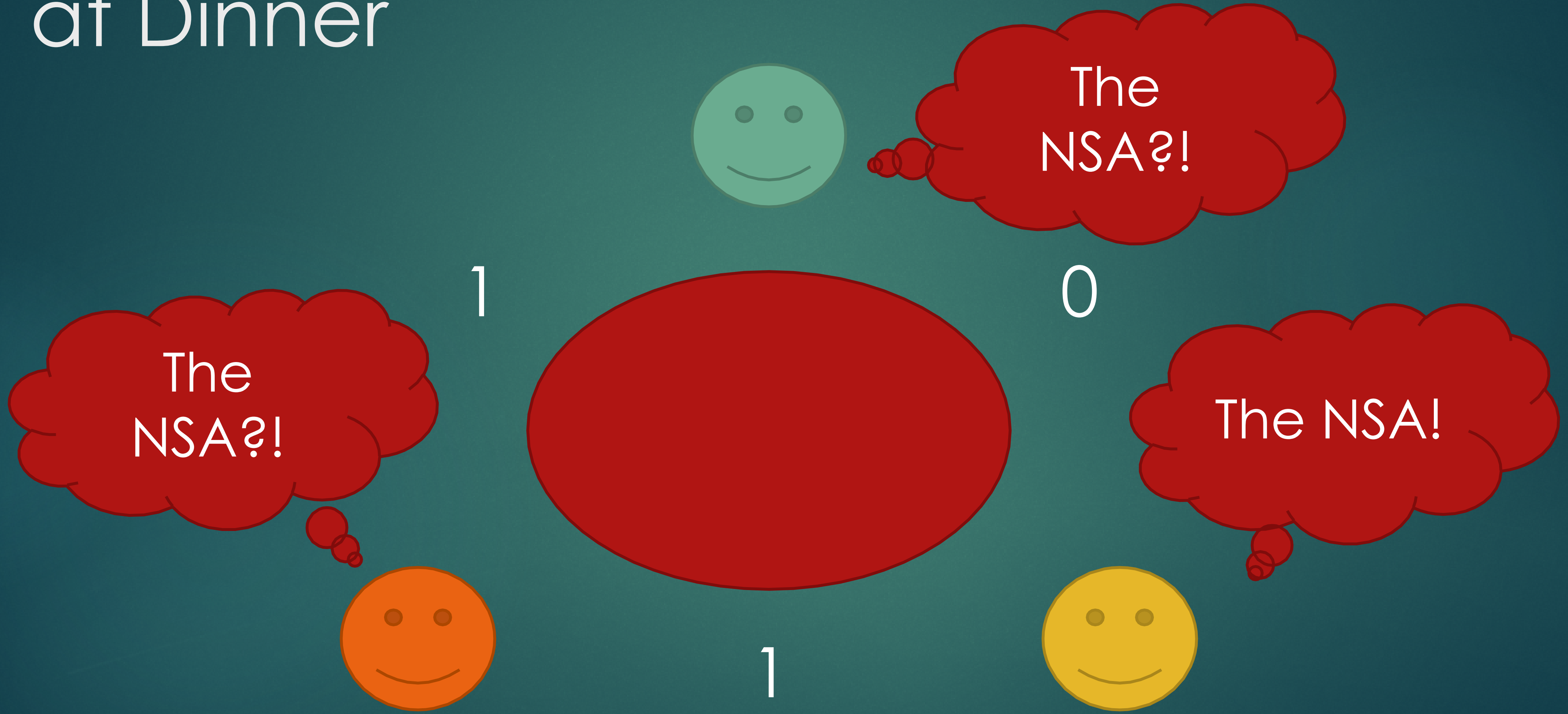
Problem 1 – Collision of Messages



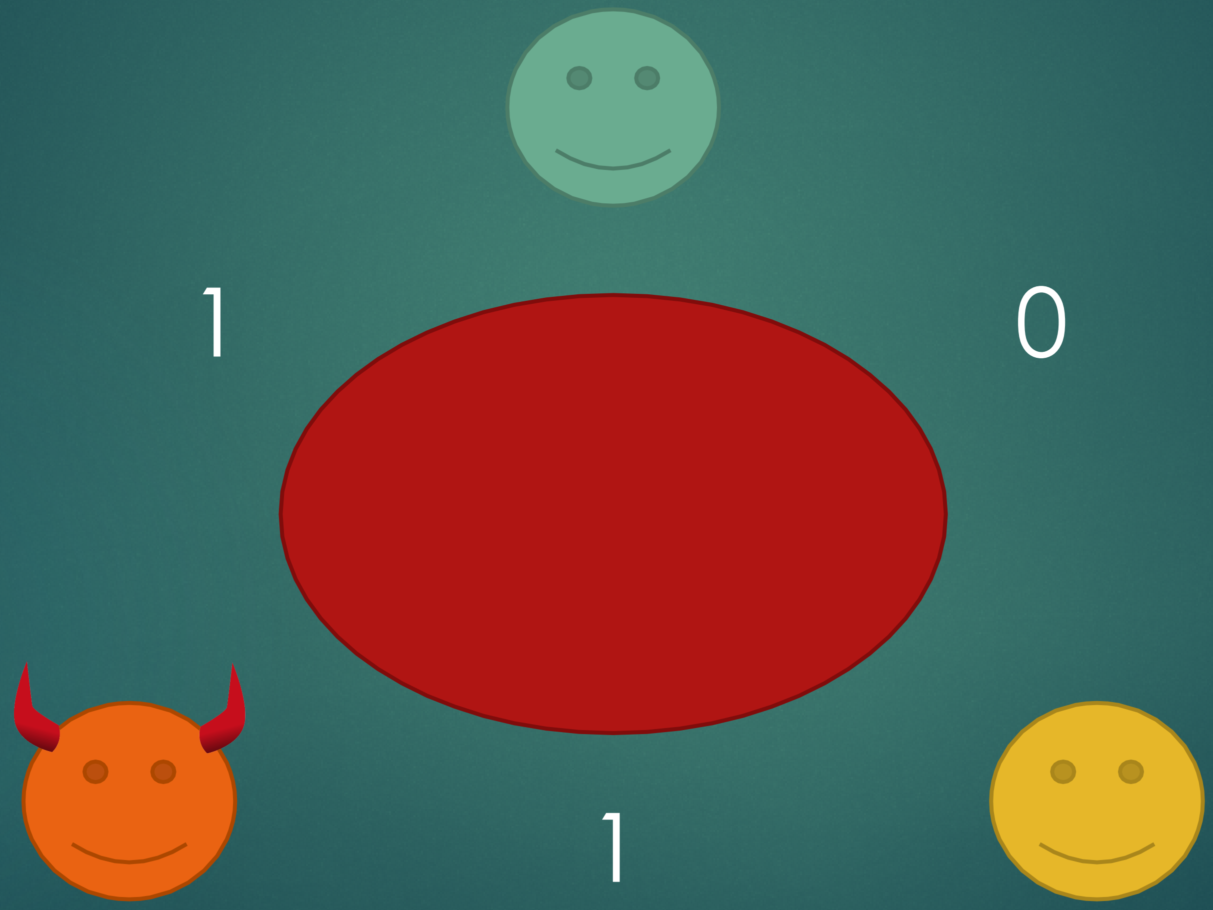
Problem 1 – Collision of Messages



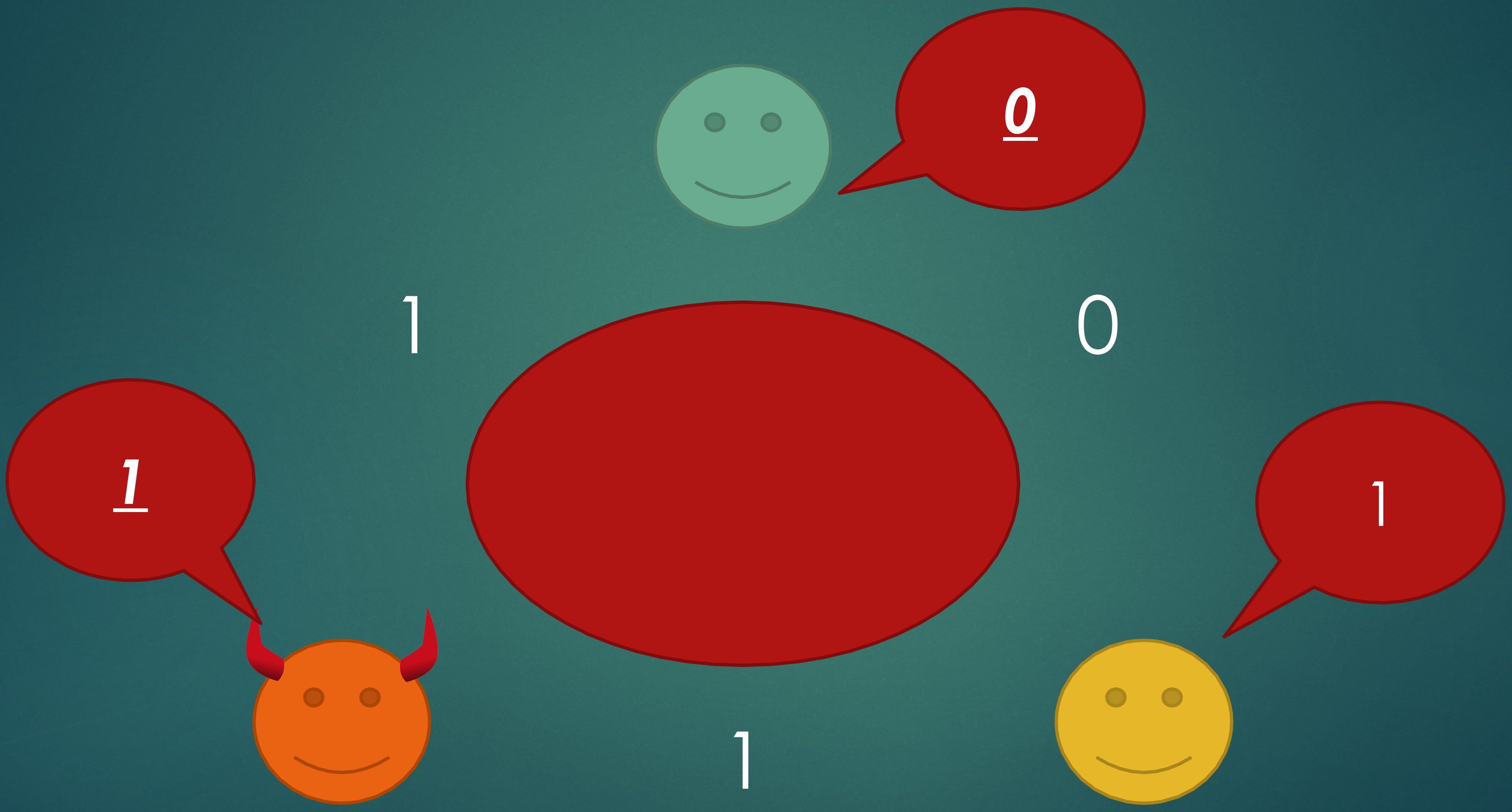
The Premise – The Cryptographers at Dinner



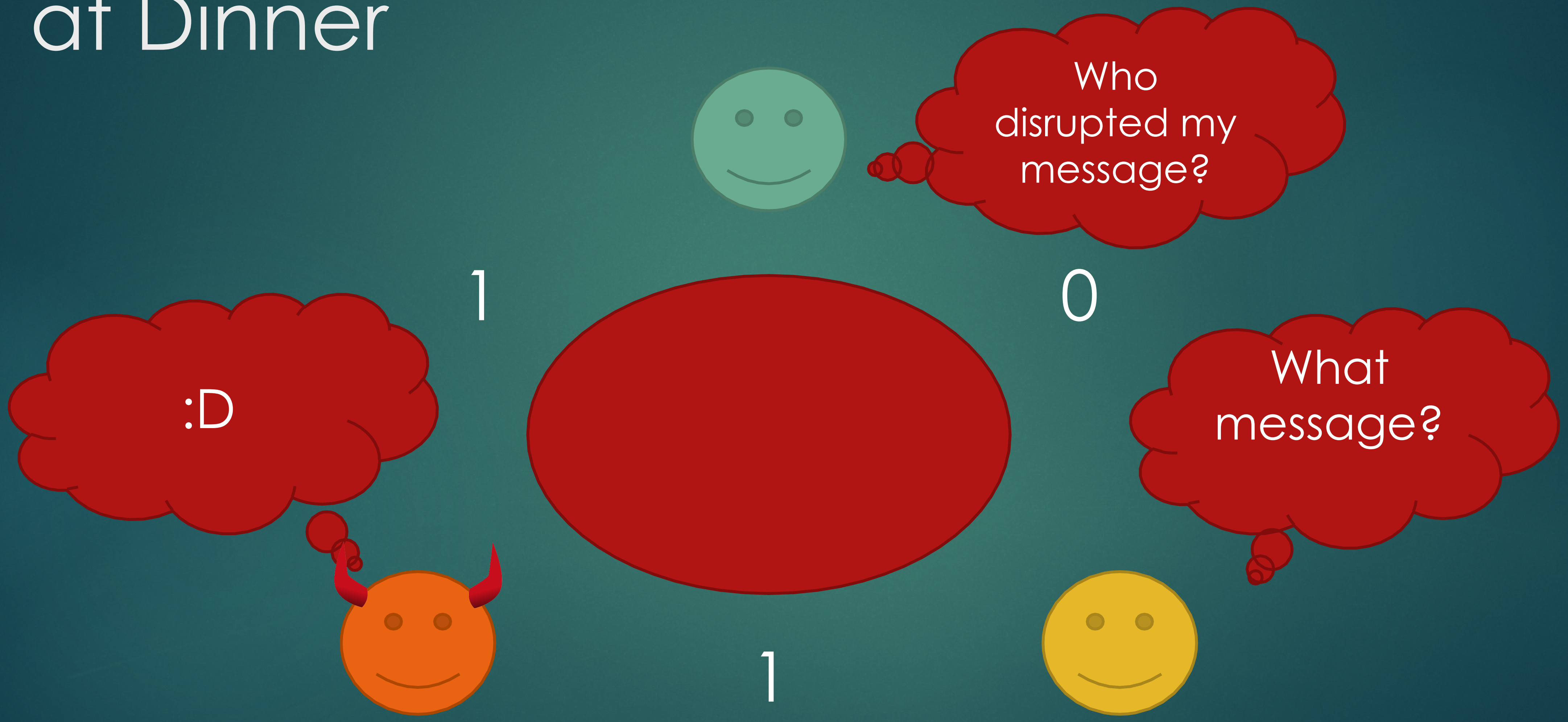
Problem 2 – Kicking Disruptors



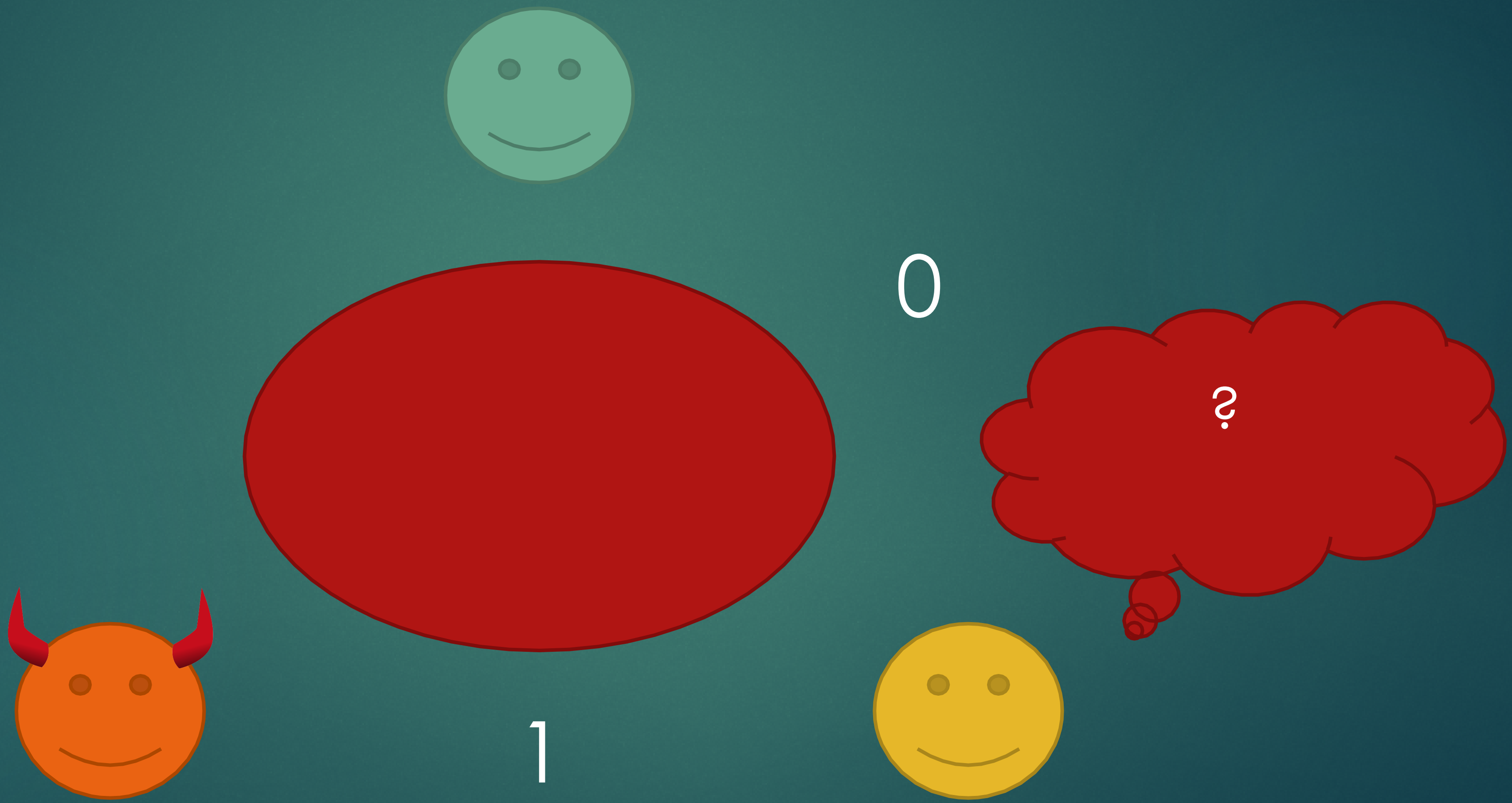
Problem 2 – Kicking Disruptors



The Premise – The Cryptographers at Dinner



Problem 2 – Kicking Disruptors



Coinshuffle ++

1. We conceptualize **P2P mixing as a natural generalization of DC-Nets**. A P2P mixing protocol enables a set of mutually distrusting peers to publish their messages simultaneously and anonymously without requiring any trusted or untrusted third-party anonymity proxy. (pg. 2)
2. DiceMix Protocol: DiceMix builds on the original DC-net protocol. P2P Mixing Protocol handles collisions by redundancy, and disruption by revealing session secrets to expose malicious peers. **DiceMix requires only 4+2*f* rounds in the presence of *f* malicious peers**, i.e., only four rounds if every peer behaves honestly
3. CoinShuffle++ Protocol: Applying DiceMix to Bitcoin transactions to create decentralized and anonymous CoinJoins.
4. Generic Attack on P2P Mixing Protocols

DiceMix – A Better DC Net Protocol

Instead, we follow the paradigm of handling collisions by redundancy [15], [19], [25], [38], [50]. Assume that messages to be mixed are encoded as elements of a finite field $\mathbb{F}$ with $|\mathbb{F}| > n$, where n is the number of peers. Given n slots, each peer i, with message m_i, publishes m_i^j (i.e., m_i to the j-th power) in the j-th slot. This yields an intentional collision involving all peers in each of the slots. Using addition in $\mathbb{F}$ instead of XOR to create DC-net messages, the j-th slot contains the power sum $S_j = \sum_i m_i^j$.

Now, we require a mechanism to extract the messages m_j from the power sums S_j. Let $g(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0$ be a polynomial with roots $m_1, m_2, \ldots, m_n$. Newton's identities [33] state

$$
\begin{aligned}
a_n &= 1, \\
a_{n-1} &= S_1, \\
a_{n-2} &= (a_{n-1} S_1 - S_2)/2, \\
a_{n-3} &= (a_{n-2} S_1 - a_{n-1} S_2 + S_3)/3, \\
&\vdots
\end{aligned}
$$

DiceMix – A Better DC-Net Protocol

1. Key Exchange ***(KE)*** : Diffie-Hellman Key-exchange between participants
2. Commitment ***(C)*** : Commitment phase to Message
3. DC-Net ***(DC)***: DC-Net protocol using power-sums over a finite field
4. Confirmation ***(CF):*** The end of a successful mix – messages are available
5. Reveal Secret Key ***(SK):*** Ephemeral keys revealed
6. Reveal Pads ***(RV):*** Shared Secrets Revealed

DiceMix – A Better DC Net

Runs	**Communication rounds**							
1	KE	CM	DC	SK				
2			KE	CM	RV DC	CF		
3					KE	CM	RV DC	CF
4							KE	CM

Fig. 2: Example of a DiceMix Execution. Run 1 fails due to DC-net disruption. Run 2 fails to confirm. Run 3 finally succeeds, and run 4 is then aborted. Rows represent protocol runs and columns represent communication rounds. Blue parts are for concurrency; the arrows depict the dependency between runs, i.e., when a run notifies the next run about the peers to exclude. `KE`: Key exchange; `CM`: Commitment; `DC`: DC-net; `RV`: Reveal pads; `SK`: Reveal secret key; `CF`: Confirmation.

DiceMix – A Better DC Net

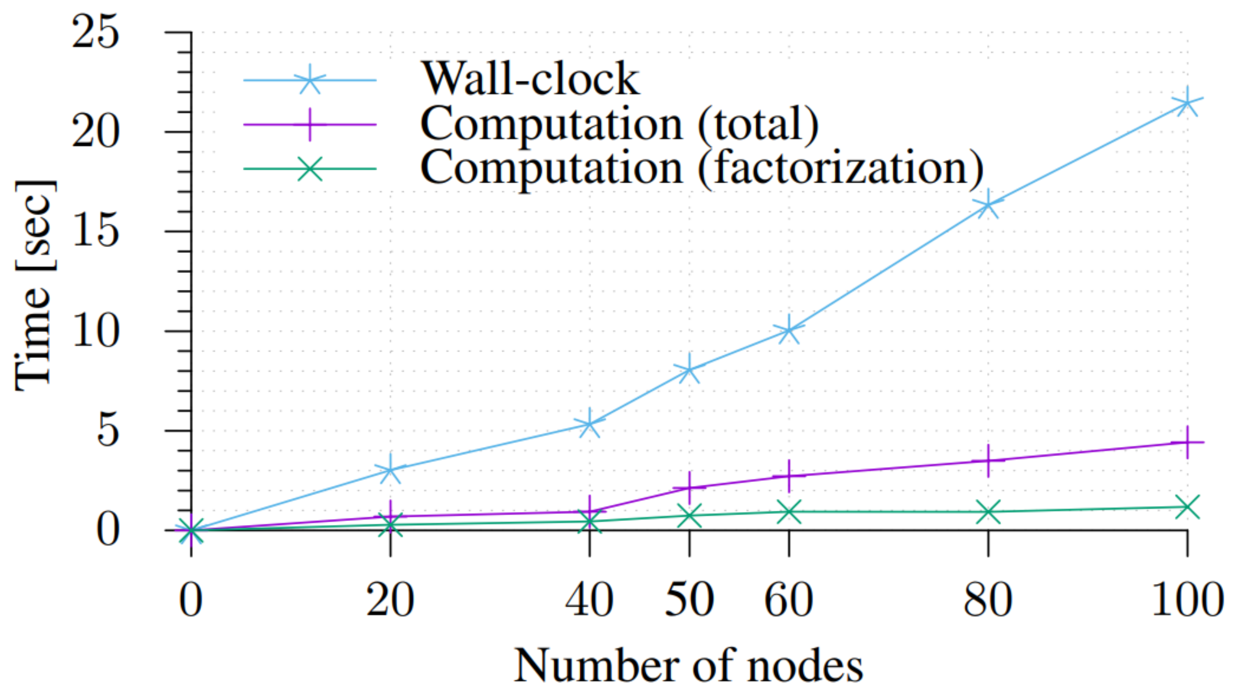


Fig. 4: Wall-clock time and computation times. All peers have a bandwidth of 10 Mbit/s; the bulletin board has a total of 1 Gbit/s; all links have 50 ms latency.

Coinshuffle ++ Summary

1. By using the DiceMix protocol instead of the original DC-Net, we can achieve **guaranteed finality** of the message protocol in **4+2f rounds**, where users anonymously post their equal-output fresh addresses. Collisions and disruption are both effectively dealt with.
2. This **protocol can scale efficiently** to allow for 50+ participants without computation costs or rapidly increasing time cost.
3. The authors claim that is a substantial improvement to CoinShuffle, which required users to **pass encrypted messages sequentially** across the entire set of participants, and thus scaled very poorly.

Questions

Questions

In an interview, Pedro Moreno-Sanchez said that the CashShuffle implementation of CoinShuffle was not correctly written. In what way, and is this still the case today?

Questions

In what way is CoinShuffle++ more desirable to ZeroLink?

Questions

DiceMix uses power sets to construct messages. Can you explain how this works and why it was chosen as a mechanism to avoid collisions?

Could you offer an example for us to observe the process?

Questions

In every failed round of DiceMix, the users must reveal which message they sent, does this mean that addresses can never be used again in this protocol? If so, this presents challenges for wallets that are recovered.