James Mickens' *The Saddest Moment*

Whenever I go to a conference and I discover that there will be a presentation about Byzantine fault tolerance, I always feel an immediate, unshakable sense of sadness ...

"How can you make a reliable computer service?" the presenter will ask in an innocent voice before continuing, "It may be difficult if you can't trust anything and the entire concept of happiness is a lie designed by unseen overlords of endless deceptive power." The presenter never explicitly says that last part, but everybody understands what's happening.

CS 168: Blockchain and Cryptocurrencies

The Blockchain and Advanced Transactions

Prof. Tom Austin

San José State University

Lab Review

GRADED Reading Summary

Before next class, read Bonneau et al.'s "*SoK: Research Perspectives and Challenges for Bitcoin and Cryptocurrencies*".

Be prepared to discuss next class.

http://www.jbonneau.com/doc/BMCNKF15-IEEESP-bitcoin.pdf

Byzantine General's Problem
(Lamport et al. 1982)

- Generals decide whether to attack a city.
- If all (or most) attack, victory is sure.
- If insufficient forces attack they will be crushed.

Complications

- Some of the generals are traitors.
  - Fake messages.
  - Lie about messages received.
- Messengers may be killed.
- Should you confirm messages?
- Should you confirm the confirmations?
- Should you confirm the confirmations of the confirmations?

Traitorous Lieutenant
(from Lamport et al. 1982)

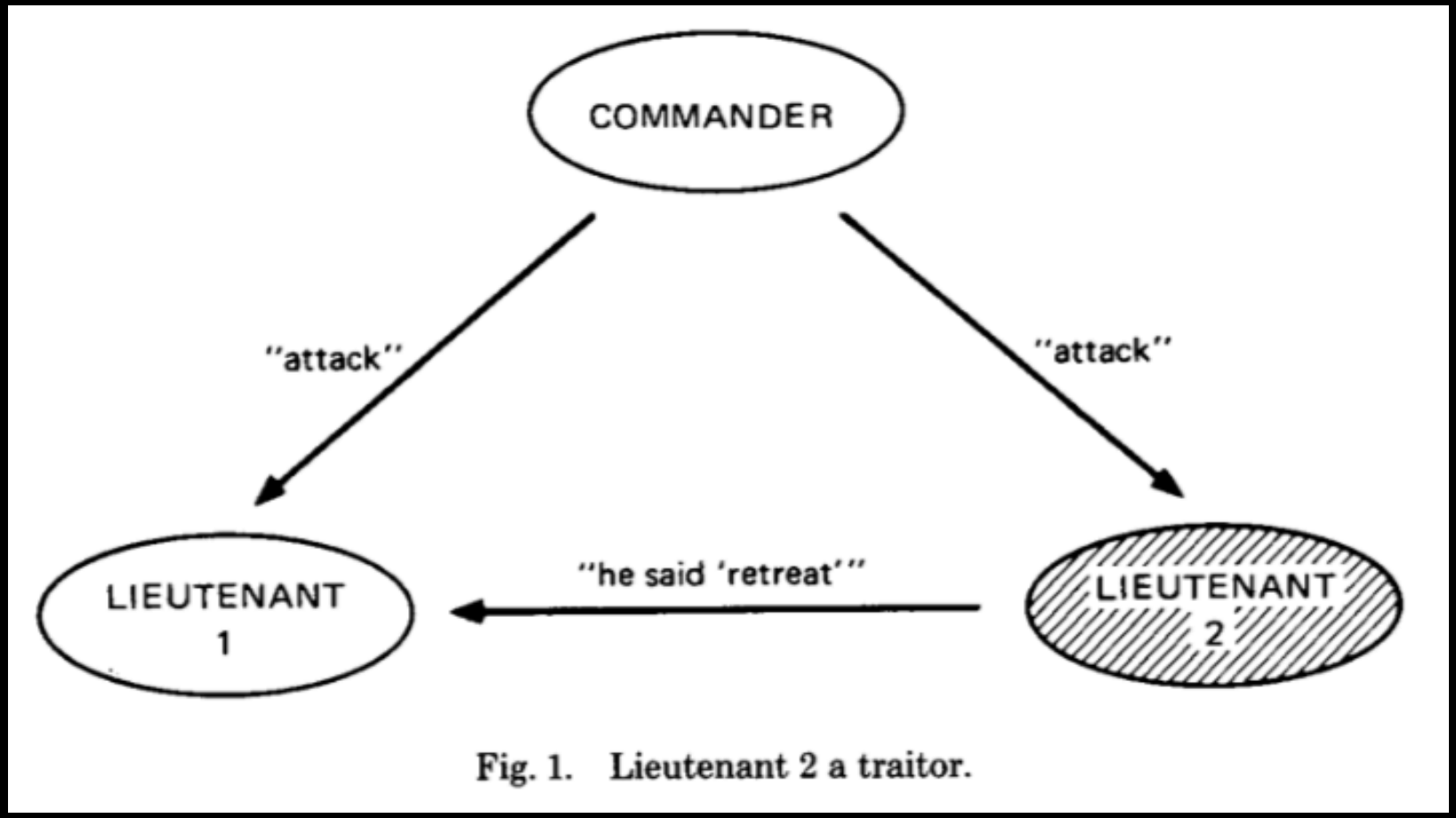

Fig. 1. Lieutenant 2 a traitor.

Traitorous General
(from Lamport et al. 1982)

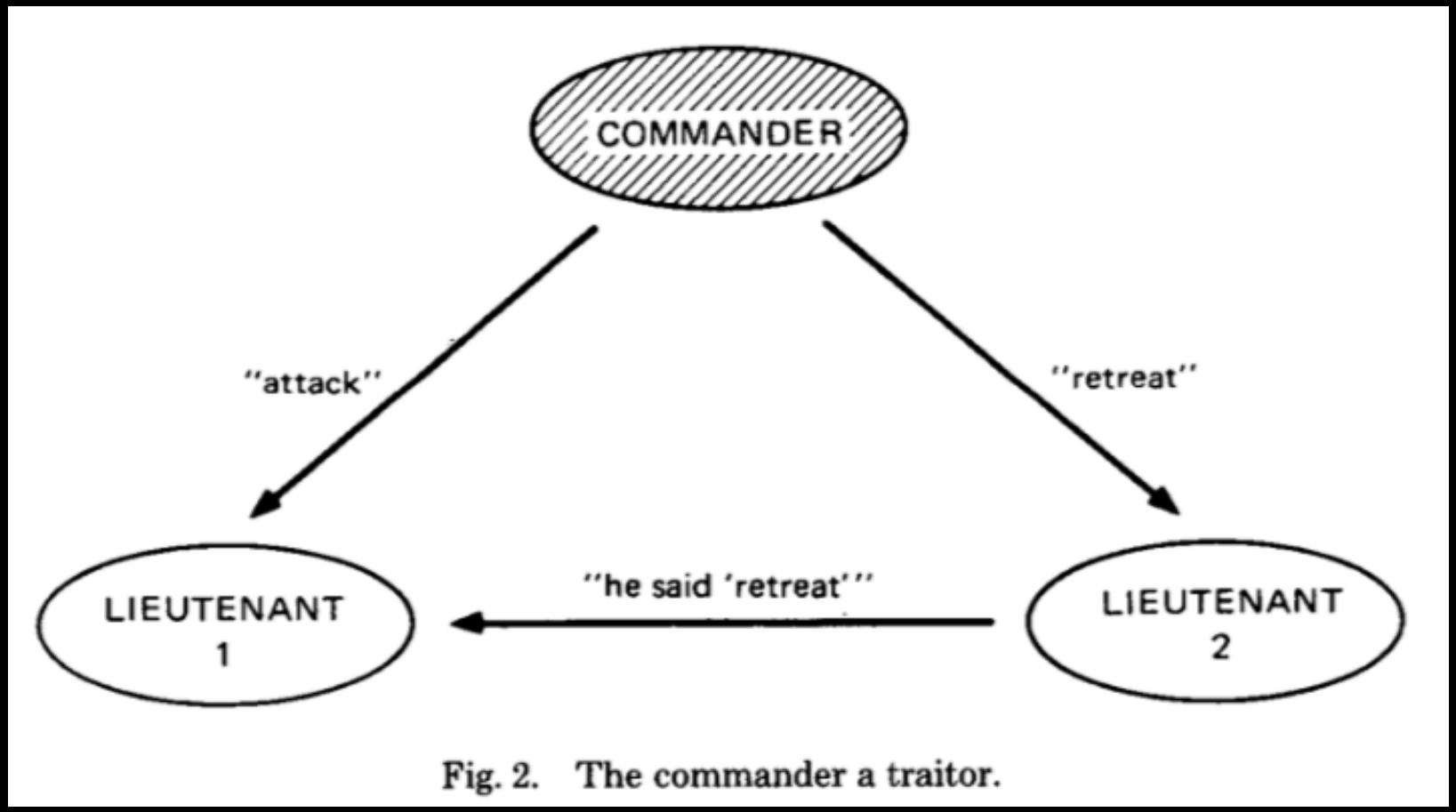

Fig. 2. The commander a traitor.

Previous Solutions for Byzantine Fault Tolerance

- Paxos
  - Provides distributed consensus
  - Formal safety guarantees
  - Complex
- Raft
  - Simplified version of Paxos
  - Also makes formal safety guarantees
  - Still complex

Distributed Consensus

- Anyone could lie to us.
- With crypto-currencies, everyone is *motivated* to lie.

So what *can* we trust?

Greed

Nakamoto Consensus

- *Probabilistic*
- One CPU, one vote (Hah!)
- Open membership
- Solving PoW puzzle determines "leader"
  - i.e., who gets to make a block.
- More profitable to mine than to cheat

What is a blockchain

- Data
- Cryptographic hash linking blocks
- Sybil resistance

Block Information

- Version
- Timestamp
- Previous block hash
- Proof-of-work (PoW) target
- Nonce (the "proof")
- Merkle root

Block Information

- Version
- Timestamp
- Previous block hash
- Proof-of-work (PoW) target
- Nonce (the "proof")
- **Merkle root**

Block Information

- Version
- Timestamp
- **Previous block hash**
- Proof-of-work (PoW) target
- Nonce (the "proof")
- Merkle root

Block Information

- Version
- Timestamp
- Previous block hash
- Proof-of-work (PoW) target
- **Nonce (the "proof")**
- Merkle root

The Sybil resistance

What data structure is a blockchain?

- A tree
  - With mostly dead branches
    - Orphan blocks
  - The surviving branches have names:
    - Bitcoin Cash, Bitcoin Gold, etc.
- A chain of "finalized" blocks
- A tree of remaining possibilities

Blockchain Properties

- Distributed
- Decentralized
- Immutable history
- Time-stamping

Blockchain Limitations

- Establishes agreed-upon ordering *not necessarily* the original ordering.
- Nasty stuff stored on the blockchain.
- Assumes actors are
  - Rational
  - Incentivized by profit

Irrational Actor

Nation-states

Lab: Blockchain Logger

Details in Canvas and on course website.

How Bitcoin *Really* Works

- SpartanGold covers most common cases
- Bitcoin is programmable
  - Supports variety of transaction types …
  - … and some non-transaction types.

Transaction Lifecycle

1. Origination: creation of a transaction.
   - All information is public.
2. Signed (one or more parties) to authorize spending.
3. Validated and propagated by nodes in network.
   - Avoid propagating invalid transaction.
4. Verified by mining node and included in block.

Transaction

- Amount of BTC, in satoshis.
  - Satoshi = 0.00000001 BTC, smallest unit of BTC.
- Locking script.
  - Terms for funds to be released.
  - Spending conditions.
    - AKA *encumbrences*.

Transaction inputs and outputs

- Mapping of inputs to outputs forms *transaction chains*.
- Inputs – stored on blockchain.
- Outputs (UTXOs) stored in RAM.

Transaction input

- Pointer to UTXO
- Unlocking script matches conditions of UTXO's locking script
- Orphan transactions
  - Transactions whose UTXO has not yet been seen by the miner.
  - Max size to prevent DoS attacks

Bitcoin "Script" Language

- Forth-like
  - Reverse-Polish notation
  - Stack based
- Lock: `scriptPubKey`
- Unlock: `scriptSig`
- https://en.bitcoin.it/wiki/Script

Sample Transaction Output

```
"vout": [
   { "value": 0.01500000,
     "scriptPubKey": "OP_DUP OP_HASH160
         ab6802… OP_EQUALVERIFY
         OP_CHECKSIG" },
   { "value": 0.08450000,
     "scriptPubKey": "OP_DUP OP_HASH160
         7f9b1a… OP_EQUALVERIFY
         OP_CHECKSIG" },
]
```

Script Limitations

- No loops.
- No complex control flow.
- Not Turing complete.
- No division.

Sample Script

```
2 7 OP_ADD 3 OP_SUB 1 OP_ADD 7 OP_EQUAL
```

(in-class)

Standard Transactions

- Safe "templates" for transactions.
- Many miners will ignore non-standard transactions.
- Non-standard transactions can still be accepted.
  - Find a miner who will accept it.

Standard Transactions

- Pay-to-public-key
- Pay-to-public-key-hash (P2PKH)
- Multi signature
  - Multiple signatures required.
  - M-of-N schemes often used.
- Pay-to-script-hash (P2SH)
- Data output

Pay-to-public-key and P2PKH

- Pay-to-public-key:
  - Unlock by presenting public key and signature
  - Public key is large
  - Allowed, but largely fading from use
- P2PKH
  - Instead uses the *hash* of the key, in hex format
  - Otherwise, works the same way

Sample P2PKH Script

```
<sig> <PubK> DUP HASH160 <PubKHash>
                    EQUALVERIFY CHECKSIG
```

(in-class)

Data Output

- Store data on the blockchain in an *unspendable* UTXO.
  - Controversial, since UTXOs are stored in RAM.
  - Blockchain bloat.
- OP_RETURN.
  - Special opcode that produces a *provably* unspendable UTXO.
    - UTXO does not need to be stored in RAM.

P2SH

- Pay to a script matching a hash.
- Creates reusable scripts.