BlockDAG protocols
SPECTRE, PHANTOM

BPASE’18, Stanford

Yonatan Sompolinsky
Joint work with Aviv Zohar
Motivation

- Algorithms on graphs are fun
- Blockchain does not scale
- BlockDAG - generalization of Satoshi’s paradigm
- 144 blocks per day $\rightarrow$ 1 million blocks per day
- Proof-of-work
- Miners
- Blocks
- Block rewards
- Transactions
- Transactions fees
- Activity on-chain
- ...

[Image of person]
- Proof-of-stake
- ByzCoin [Kogias et al.], Hybrid Consensus/Thunderella [Shi, Pass]
- Lightning Network [Poon and Dryja]
- Sharding protocols
- Blockchain project X
BlockDAG paradigm
BlockDAG with 10 blocks per second
BlockDAG with 10 blocks per second
DAG mining protocol

---

rule #1: reference tip of longest chain all tips of DAG
rule #2: publish block immediately
Key observations

1. Two honest blocks unconnected in the DAG only if created at approx same time
2. Rarely more than $k$ honest blocks created at approx same time
3. Attacker can create arbitrary structures, but <50%

$k = k(D \cdot \lambda)$
$\lambda = \text{block rate}$
$D = \text{propagation delay}$
$k = 0$

$k = 2$

$k = 10$

$k \leq 2 \cdot D \cdot \lambda$

$D = \text{prop delay}$

$\lambda = \text{block rate}$
BlockDAG paradigm

- It’s a paradigm, not a solution
- Still need to resolve conflicts, aka double-spends
The PHANTOM protocol
PHANTOM’s goal: recognize cluster of honest blocks, discard (or penalize) the rest
Definition: A k-cluster is a subset of blocks such that each block in it is connected to every block in it, apart from at most k blocks.

- E not connected to D, H
- F not connected to H, I
- J not connected to E, F, K, L, D, H, I

2-cluster
Good news #1: honest blocks form a $k$-cluster

Key observations:

1. Two honest blocks are unconnected in the DAG only if created at approx same time
2. At most $k$ blocks created at approx same time

honest large $k$-cluster
Good news #1: honest blocks form a $k$-cluster
Bad news: attacker forms a $k$-cluster as well
Good news #1: honest blocks form a $k$-cluster
Bad news: attacker forms a $k$-cluster as well
Good news #2: honest blocks form a larger one!

honest large $k$-cluster

attacker small $k$-cluster
Algorithm (NP hard version):
1. Search for largest $k$-cluster
2. Order its blocks via some topological ordering
3. Iterate over blocks in the prescribed order, and accept transactions consistent with history
The PHANTOM protocol

Algorithm (NP hard version):
1. Search for largest $k$-cluster
2. Order its blocks via some topological ordering
3. Iterate over blocks in the prescribed order, and accept transactions consistent with history

why is this secure?
Generalization of Satoshi’s paradigm
Forks in the blockchain

---

deliberate attack? miner of C ignored B
propagation delay? miner of C didn’t receive B yet

What happened here? Who is honest???
Recognizing honest chain of blocks

- - -

Longest = Honest

deliberate attack? miner of C ignored B
propagation delay? miner of C didn’t receive B yet
Recognizing honest chain of blocks

why is this secure?

deliberate attack? miner of C ignored B
propagation delay? miner of C didn't receive B yet

Longest = Honest
Recognizing honest chain of blocks

Key observations
1. Two honest blocks unconnected in the DAG chain only if created at approx same time
2. Rarely more than $k \leq 1$ honest block created at approx same time
3. Attacker can create arbitrary structures, but $<50\%$
Longest chain = Largest 0-cluster
Theorem (Bitcoin):
If attacker $< 50\%$, then under low throughput, the largest 0-cluster (aka longest chain) was created by honest nodes.

Theorem (PHANTOM):
If attacker $< 50\%$, then under any throughput, the largest $k$-cluster was created by honest nodes, for some suitable $k$. 
Algorithm (NP hard version):
1. Search for largest $k$-cluster
2. Order its blocks via some topological ordering
3. Iterate over blocks in the prescribed order, and accept transactions consistent with history
Why not longest-chain of DAG??

b/c orphans do not contribute to honest chain
Why not count all orphans??

weight = 11

weight = 6
Why not count all orphans??

b/c orphans will contribute to attacker chain
PHANTOM’s solution: Count all orphans that are well connected to the chain

Longest chain: count no orphans (attackable)
Previous suggestion: count all orphans (attackable)
PHANTOM: count some orphans (secure)
Definition: A $k$-uncle of a chain is a block which is unconnected to at most $k$ blocks in it

Examples:
B - 0-uncle (it’s inside the chain)
D - 1-uncle (unconnected to $E$)
H - 2-uncle (unconnected to $E,F$)
C - 4-uncle (unconnected to $B,E,F,K$)
PHANTOM (greedy version):
1. Search for chain with largest \# of uncles of degree $\leq k$; the chain + uncles are honest
2. Order its blocks via some topological ordering
3. Iterate over blocks in the prescribed order, and accept transactions consistent with history

PHANTOM (NP hard version):
1. Search for largest $k$-cluster; the cluster is honest
2. Order its blocks via some topological ordering
3. Iterate over blocks in the prescribed order, and accept transactions consistent with history
The chain with largest number of 2-uncles:
Security guarantee: For properly chosen $k$,

1. (almost) all honest block are uncles of degree $\leq k$ of heaviest chain

2. (almost) no attacker blocks are uncles of degree $\leq k$ of heaviest chain

3. any topological order over honest blocks will converge
PHANTOM pros and cons

- throughput unlimited by the protocol
- easy to implement efficiently
- poor waiting times when conflicts are visible
## Throughput

<table>
<thead>
<tr>
<th>Desired throughput</th>
<th>Required bandwidth</th>
<th>Required storage</th>
<th>Block size</th>
<th>Block creation rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>1,000 txns per sec</td>
<td>0.5 MB per sec</td>
<td>~100 GB per day</td>
<td>50 KB</td>
<td>10 blocks per sec</td>
</tr>
<tr>
<td>10,000 txns per sec</td>
<td>5 MB per sec</td>
<td>~1 TB per day</td>
<td>100 KB</td>
<td>50 blocks per sec</td>
</tr>
<tr>
<td>1,000,000 txns per sec</td>
<td>500 MB per sec</td>
<td>~0.1 PB per day</td>
<td>0.5 GB</td>
<td>1 block per sec</td>
</tr>
</tbody>
</table>
### Scalability, SPECTRE vs PHANTOM

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Throughput limited by</th>
<th>Confirmation times</th>
<th>Ordering</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>w/o conflicts</td>
<td>w/ conflicts</td>
</tr>
<tr>
<td>Bitcoin</td>
<td>latency</td>
<td>slow</td>
<td>slow</td>
</tr>
<tr>
<td>SPECTRE</td>
<td>capacity</td>
<td>very fast</td>
<td>not guaranteed*</td>
</tr>
<tr>
<td>PHANTOM</td>
<td>capacity</td>
<td>slow</td>
<td>very slow</td>
</tr>
<tr>
<td>SPECTRE + PHANTOM</td>
<td>capacity</td>
<td>very fast</td>
<td>very slow</td>
</tr>
</tbody>
</table>

*under active balancing attack; Weak Liveness*
More stuff in the paper…

---

- Other greedy variants
- How to incorporate “red” blocks as well
- Formal proof of convergence
- Waiting times in PHANTOM
- Combining SPECTRE and PHANTOM