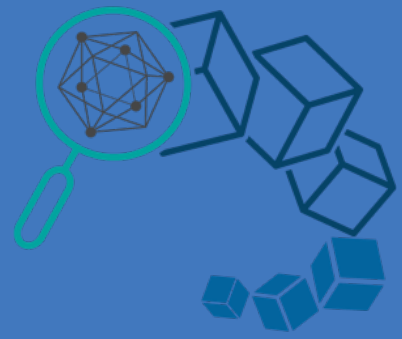


Hyperledger Fabric 1.0

Architecture Design and Application Development

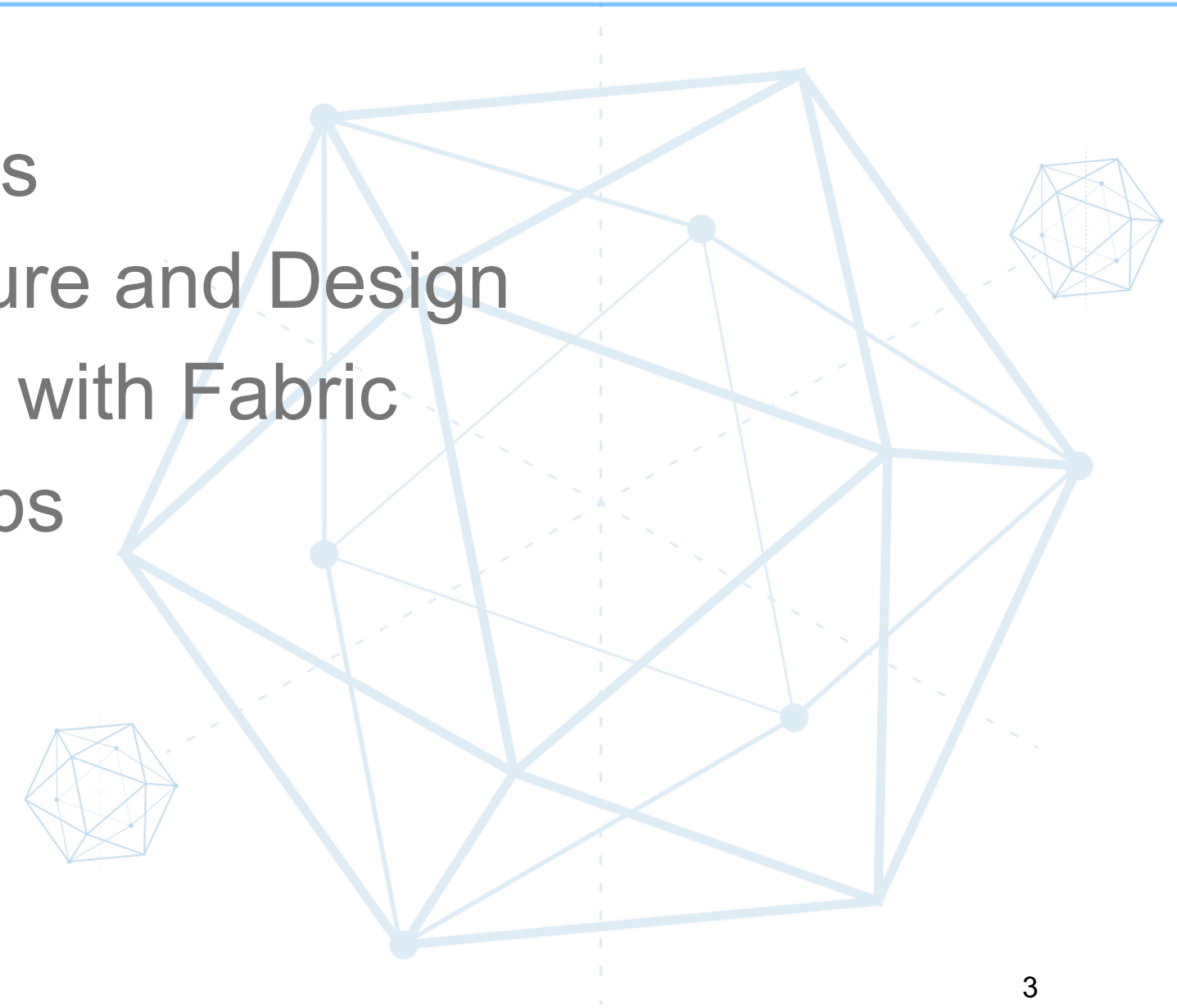
Baohua Yang
April 17, 2017

About Me

- **Researcher in IBM**
  - Fintech, Cloud and Analytics
- **Open-Source contributor**
  - Hyperledger, OpenStack, OpenDaylight, etc.
- **Hyperledger developer**
  - Code committer to fabric, sdk, Cello etc.
  - PTL of Cello project and fabric-sdk-py project
  - Chair of Hyperledger Technical Working Group China
  - Drafter of fabric sdk spec and multi-channel consensus spec

Outline

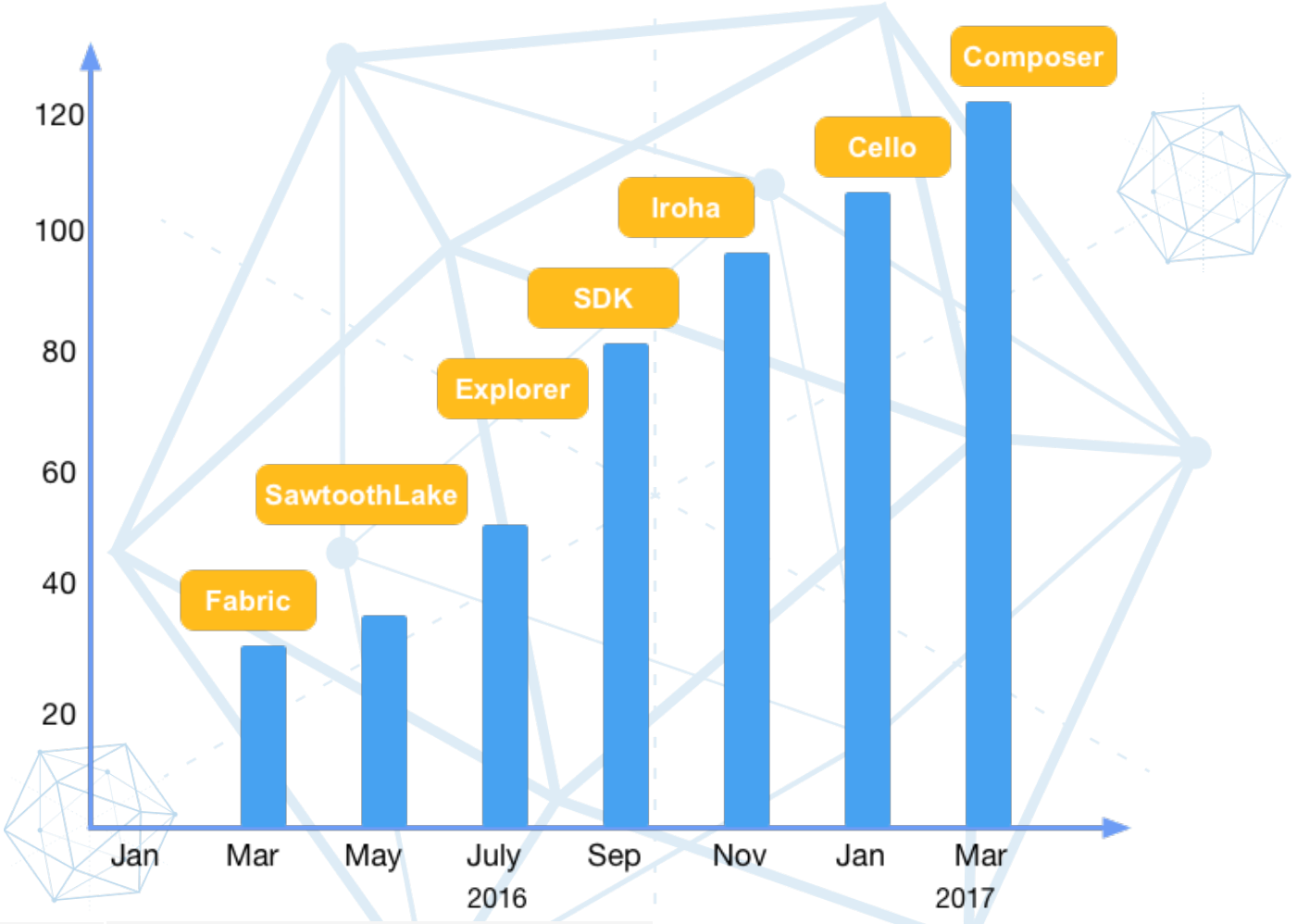
- Hyperledger Projects
- Fabric 1.0 Architecture and Design
- Develop Application with Fabric
- Deploy and Run Apps
- Q&A



Hyperledger Projects

- Since Dec 17, 2015
- Apache v2 License
- 30 founded members
- 30/129 (China) members
- 6 top projects
- 200+ contributors
- 9000+ commits

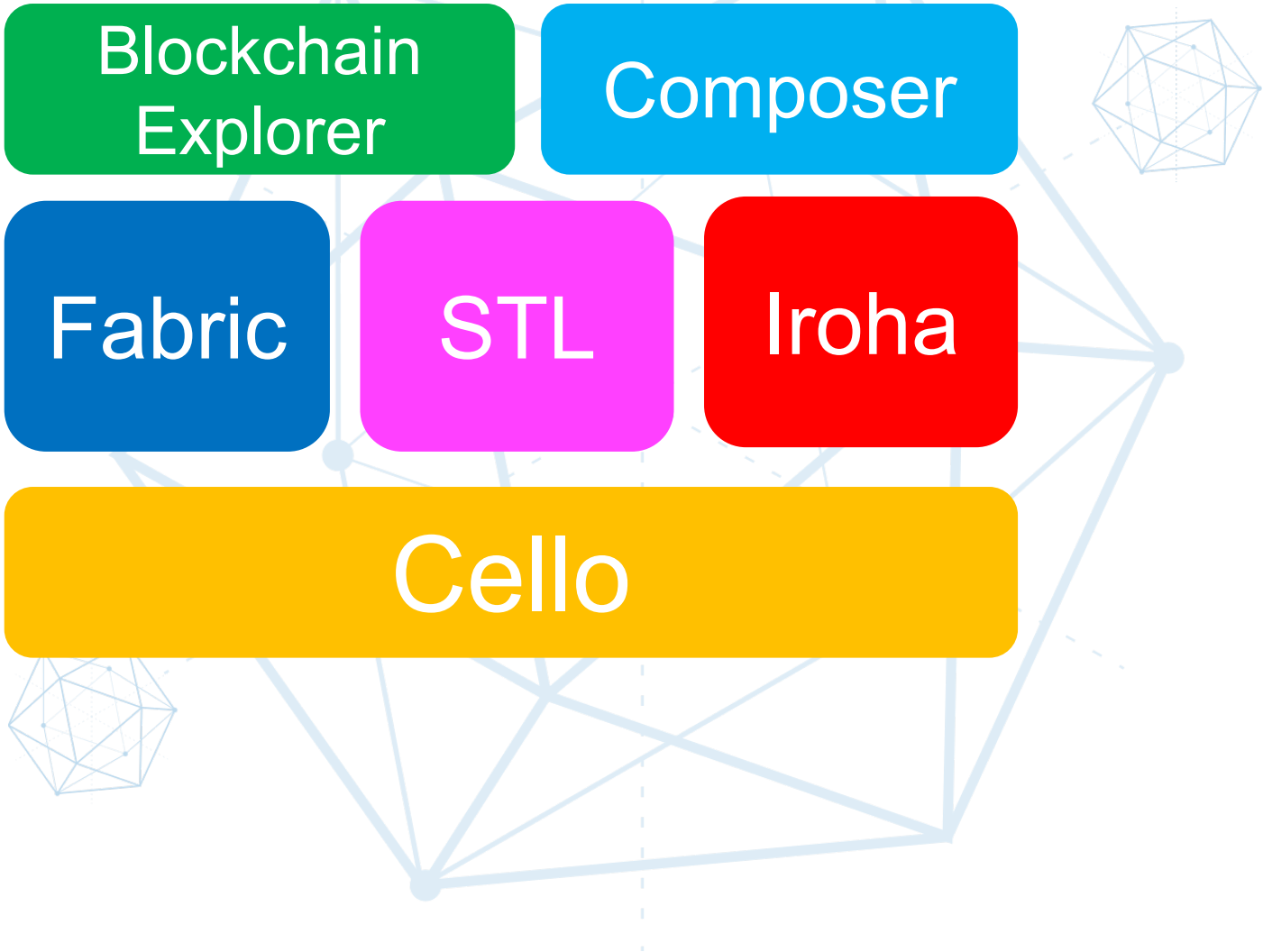
Enterprise grade, open source distributed ledger framework!



accenture High performance. Delivered. · AIRBUS · AMERICAN EXPRESS · CME Group · DAIMLER · Digital Asset · DTCC · FUJITSU · HITACHI Inspire the Next · IBM · intel · J.P.Morgan · R · SAP

Hyperledger Projects

- 6 top projects
  - Fabric
  - SawtoothLake
  - Iroha
  - Cello
  - Blockchain Explorer
  - Composer



Hyperledger Community

- Linux Foundation Support
- Organizations
  - Technical Steering Committee
  - Governing Board
  - Linux Foundation Staffs
- TWG-China
  - wiki.hyperledger.org/groups/tsc/technical-working-group-china

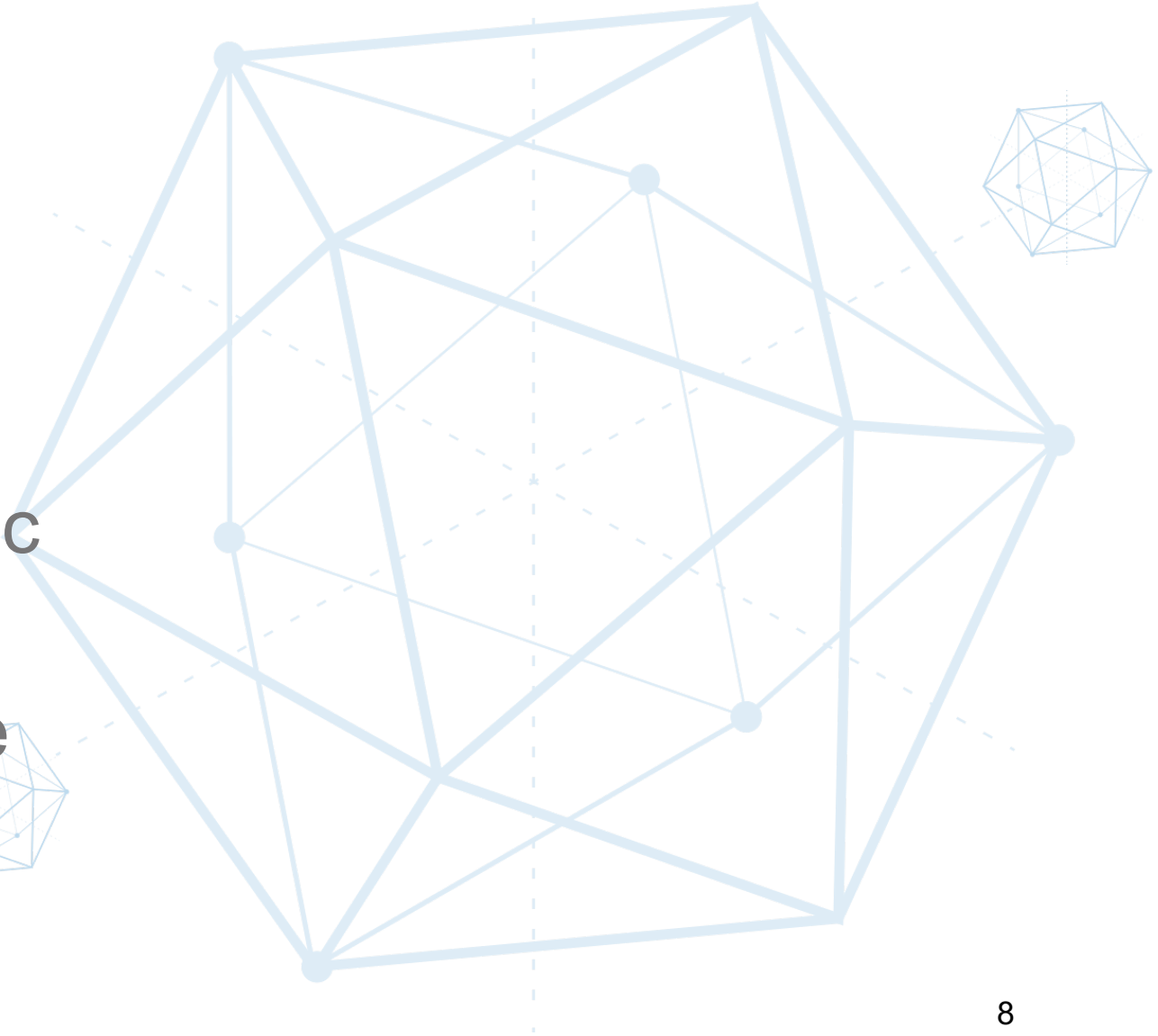
Outline

- Hyperledger Projects
- Fabric 1.0 Architecture and Design
- Develop Application with Fabric
- Deploy and Run Apps
- Q&A

Hyperledger Fabric

- Open-sourced at Dec, 2015
- Proposed by IBM and DAH
- Written in Golang
- 70+ contributors
- 4000+ commits
- v0.6: ~80k loc; v1.0: ~310k loc

- Active now, in 1.0 pre-release



Existing Blockchain Technologies

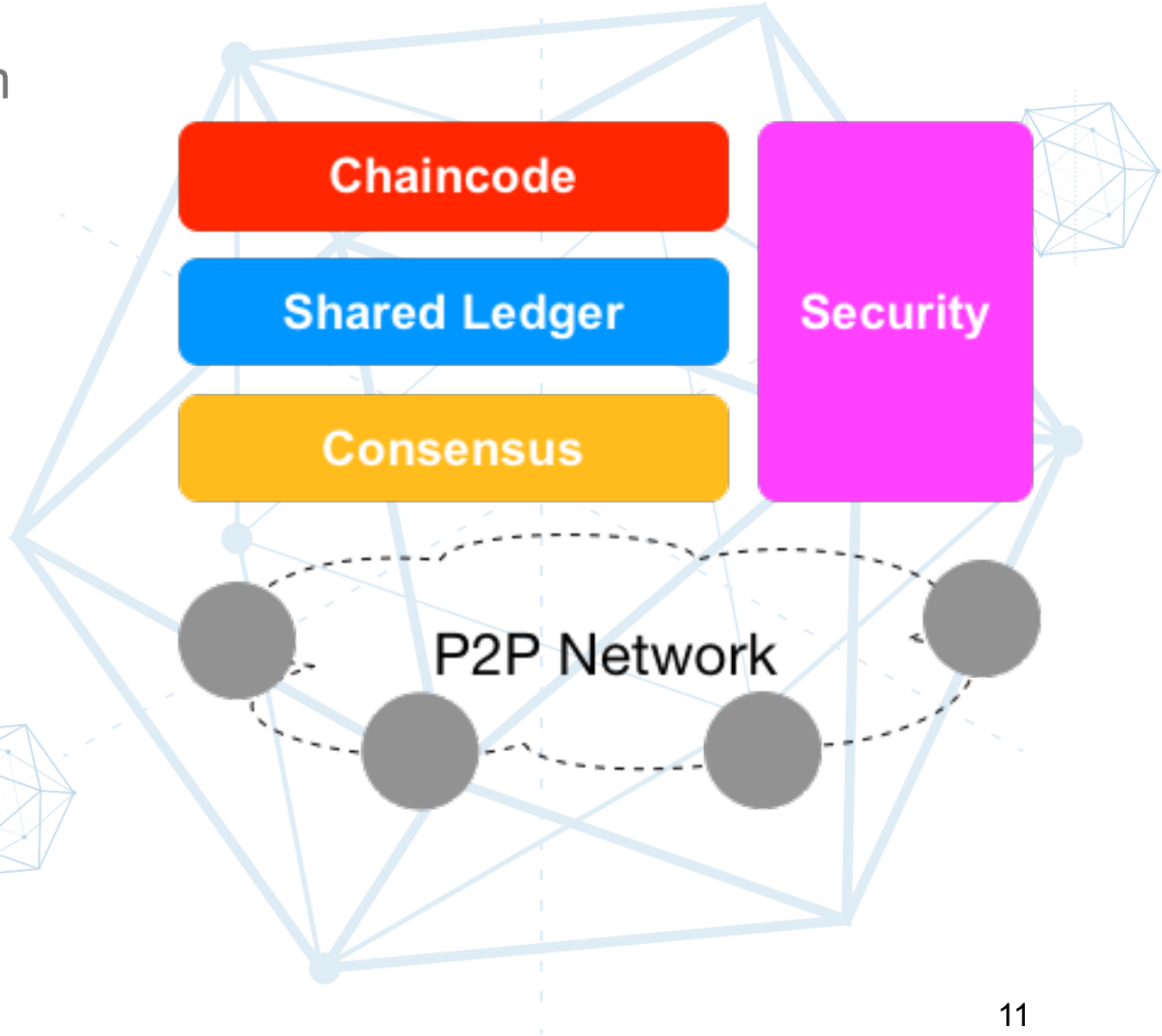
- Limited Throughput
- Slow Transaction Confirmation
- Designed for Cryptocurrency
- Poor Governance
- No Privacy
- No Settlement Finality
- Anonymous Processors
- …

Hyperledger Fabric: Ledger for Enterprise

- Privacy, Confidentiality, Auditability, Performance and Scalability
- Permissioned with better trust among members, while enable optimized consensus
- Open protocol/standard with open-source code

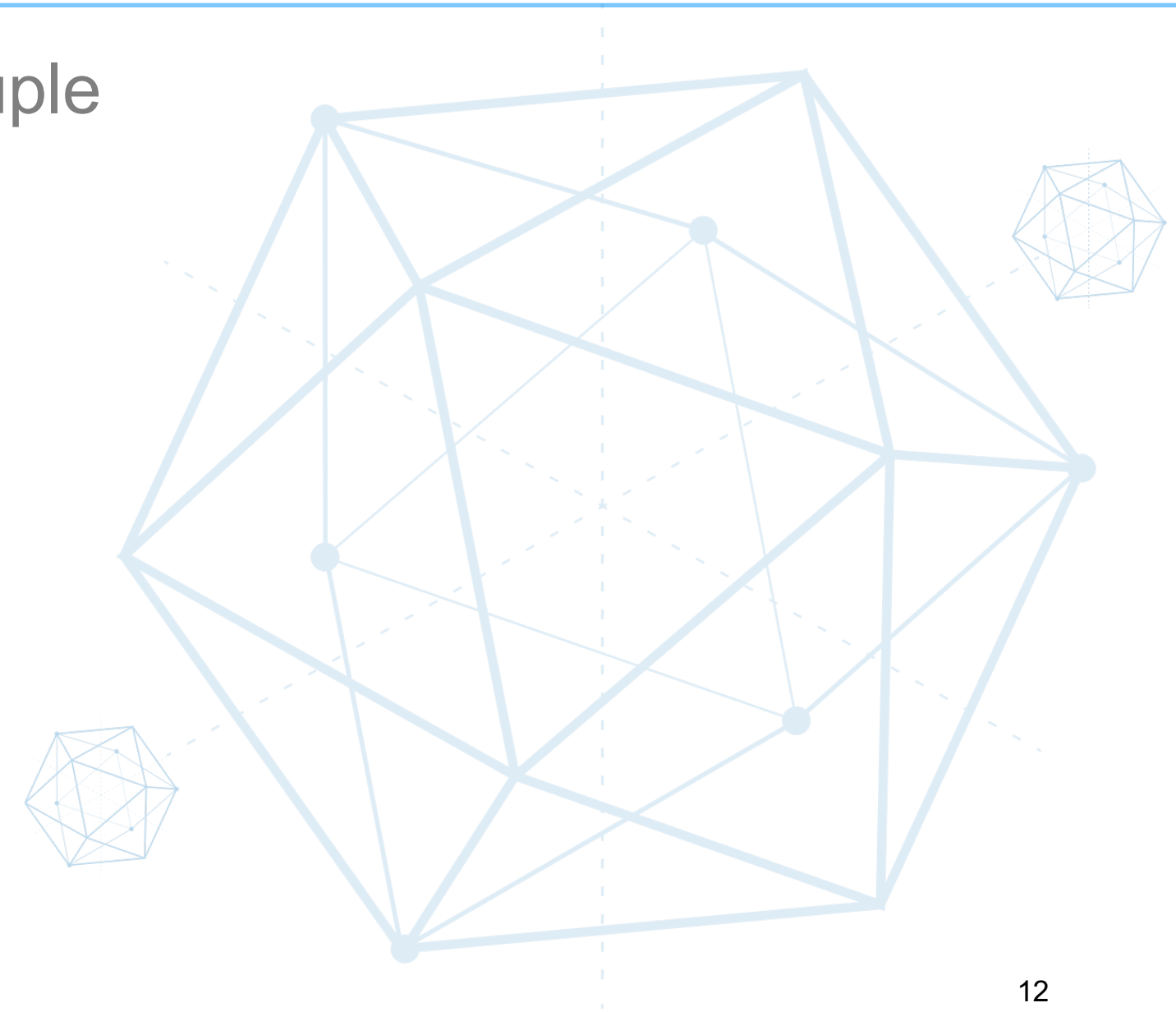
Fabric Main Components

- Shared Ledger
  - Append-only distributed record system
  - Blocks + States
- Smart Contract (Chaincode)
  - Business logics with transactions
  - Stateless and deterministic
- Consensus
  - Verified and ordered transactions
- Security
  - Access control
  - Privacy protection
  - Verification
  - CA



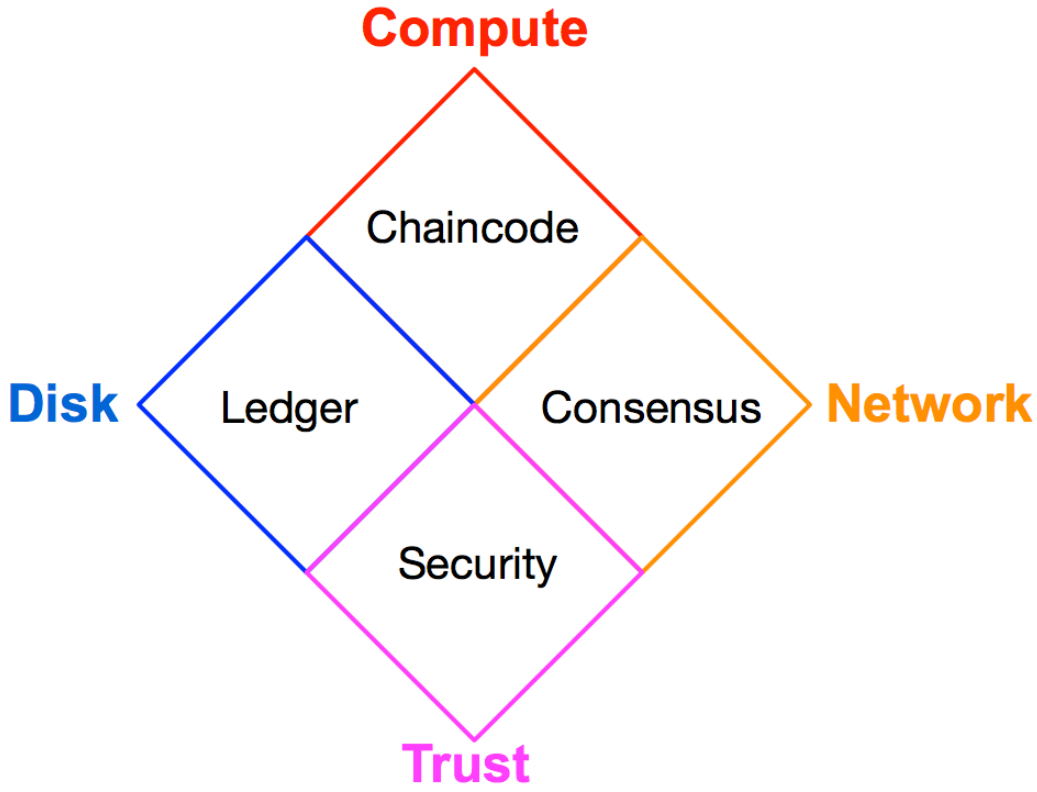
Fabric 1.0 Key Design

- Node Functionality Decouple
- Multi-Channel/Chain
- Consensus
- Permission and Privacy
- System Chaincode
- Pluggable Components



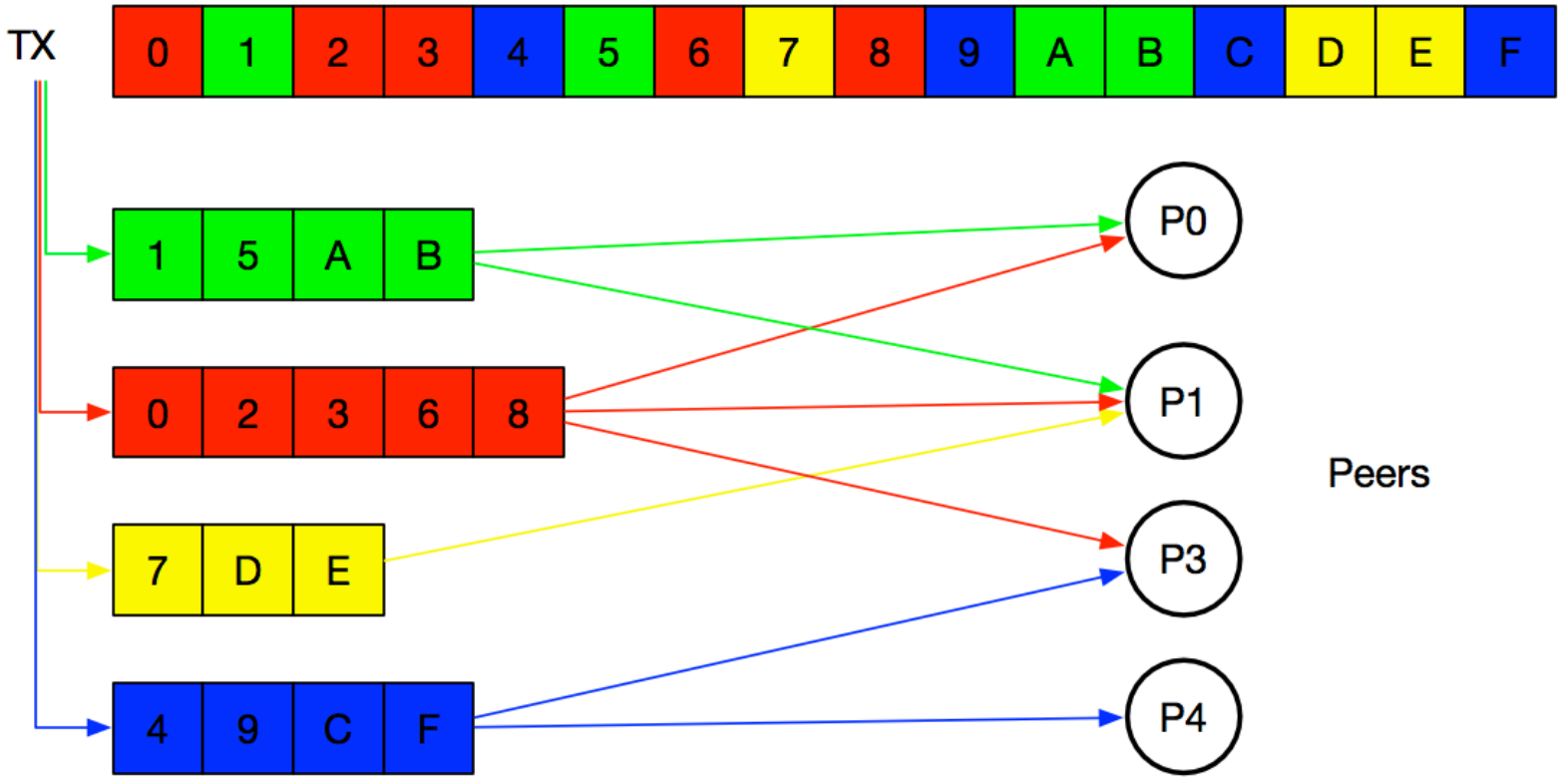
Node Functionality Decouple

- Various Intensive Requirements/Workloads
  - Chaincode: Compute intensive
  - Shared Ledger: Disk intensive
  - Consensus: Network intensive
  - Security: Trust intensive
- Decouple Full-functional Nodes
  - **Endorser**: Endorse TX proposal
  - **Committer**: Write down block
  - **Orderer**: Only order, no TX aware
  - **CA**: Certificate management



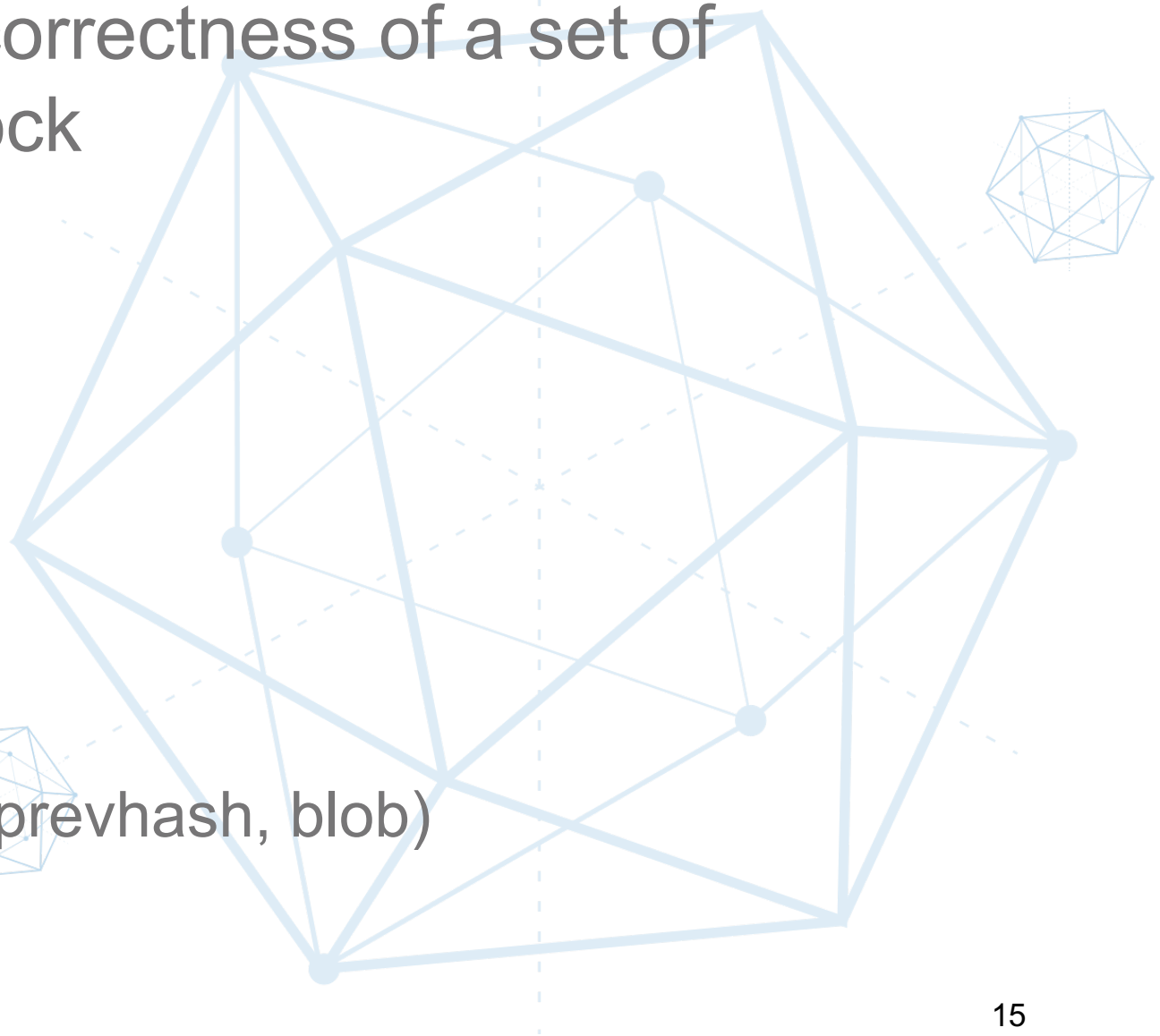
Multi-Channel/Chain

- Isolate the transactions, ledgers between organizations – Overlay Network
- Peer can join channels accordingly



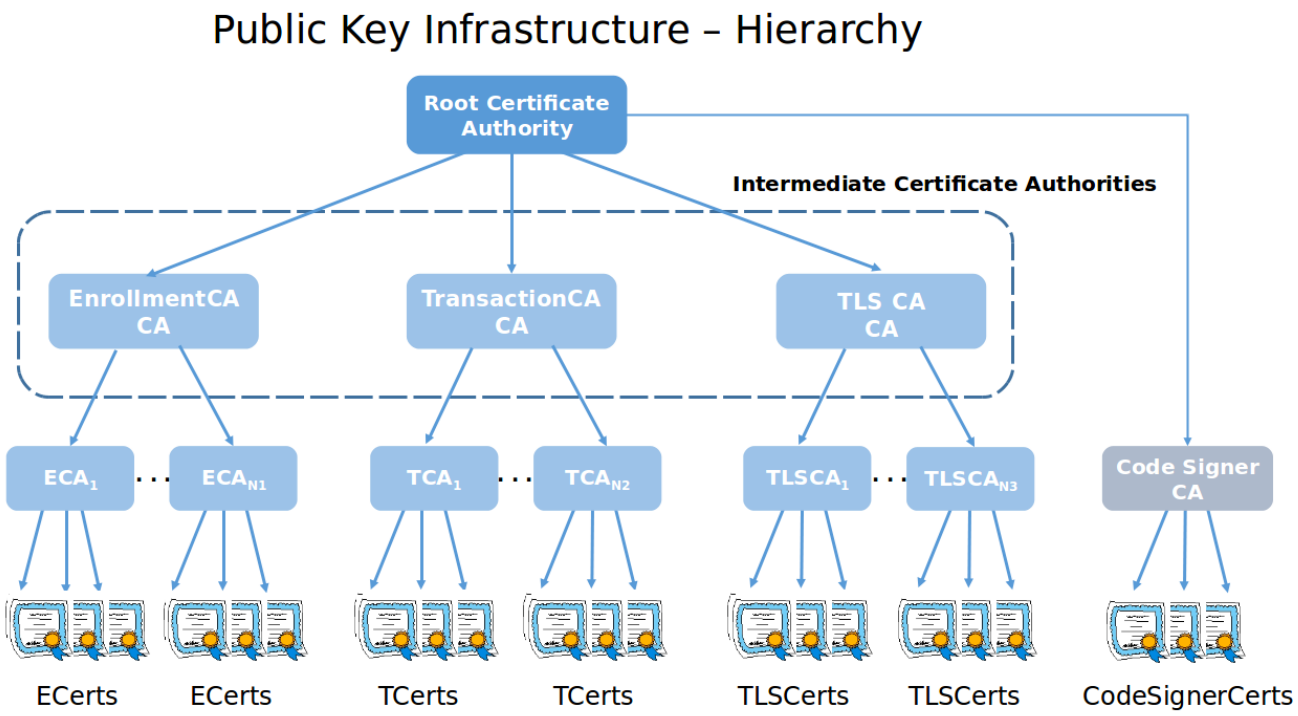
Consensus

- Full-circle verification of the correctness of a set of transactions comprising a block
  - Endorsement policy
  - MVCC validation on RW sets
  - Ordering
  - ACL
- Orderer
  - Solo, Kafka, BFT, and more…
  - Broadcast(blob), Deliver(seqno, prevhash, blob)



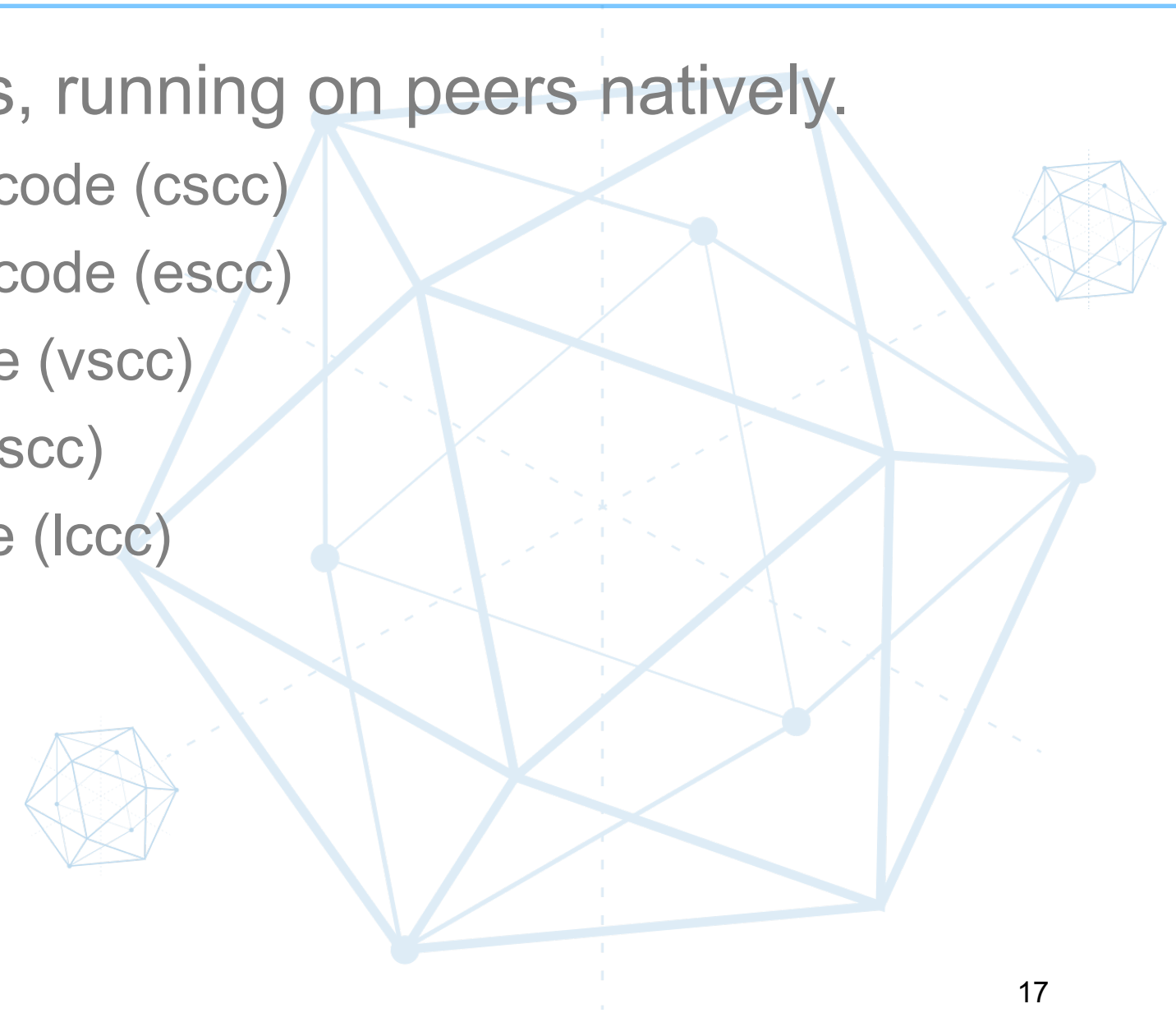
Permission and Privacy

- Permission at Various Levels
  - Network, channel, transaction
- Privacy for Business
  - Anonymity
  - Un-linkability
  - Auditability and Accountability
- Fabric CA (PKI)
  - Identity Registration Management
  - Enrollment Cert (Ecert) and Transaction Cert (Tcert)



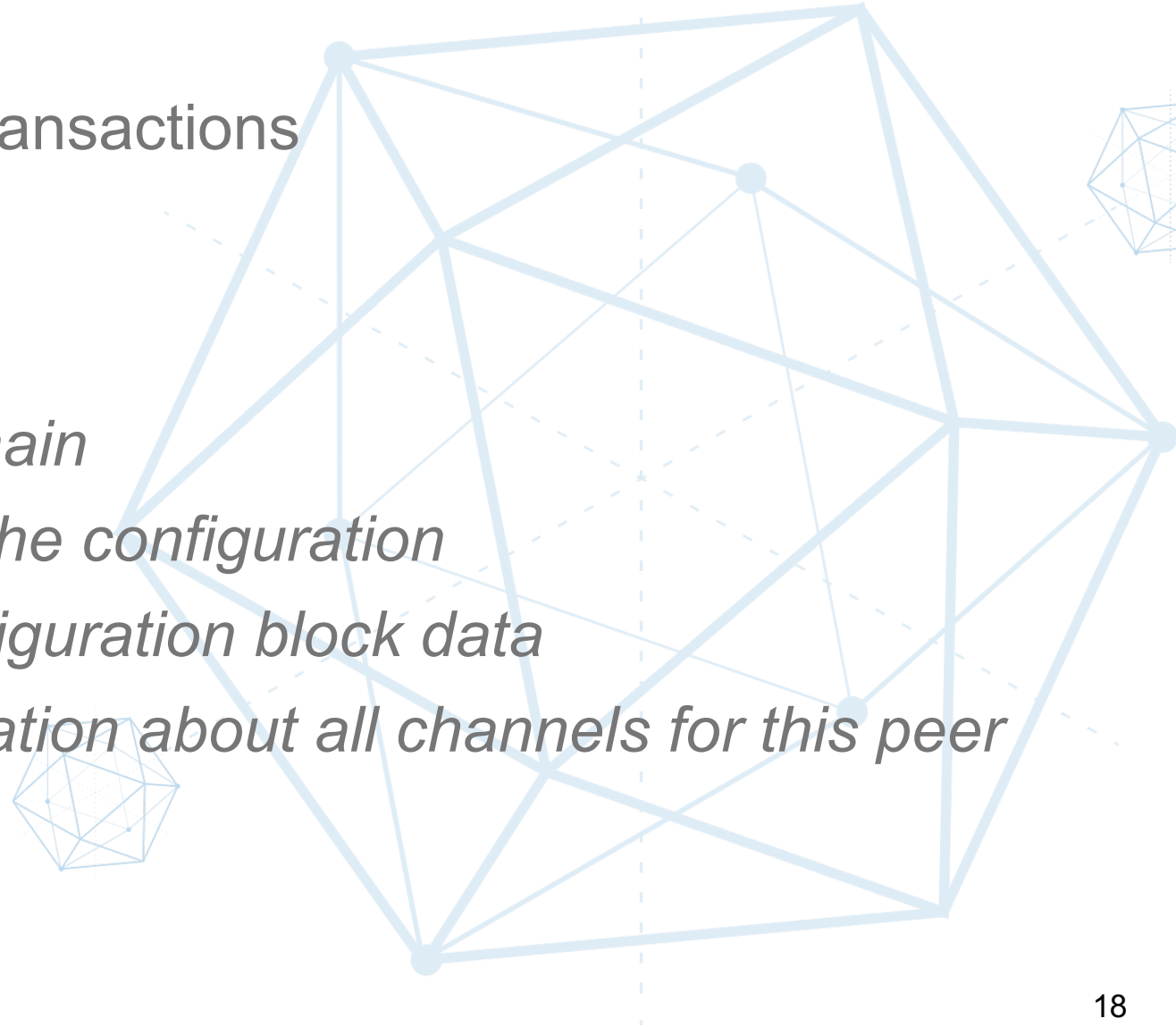
System Chaincode

- Handle system operations, running on peers natively.
  - Configuration System Chaincode (cscc)
  - Endorsement System Chaincode (escc)
  - Validation System Chaincode (vscc)
  - Query System Chaincode (qscc)
  - Life-cycle System Chaincode (lccc)



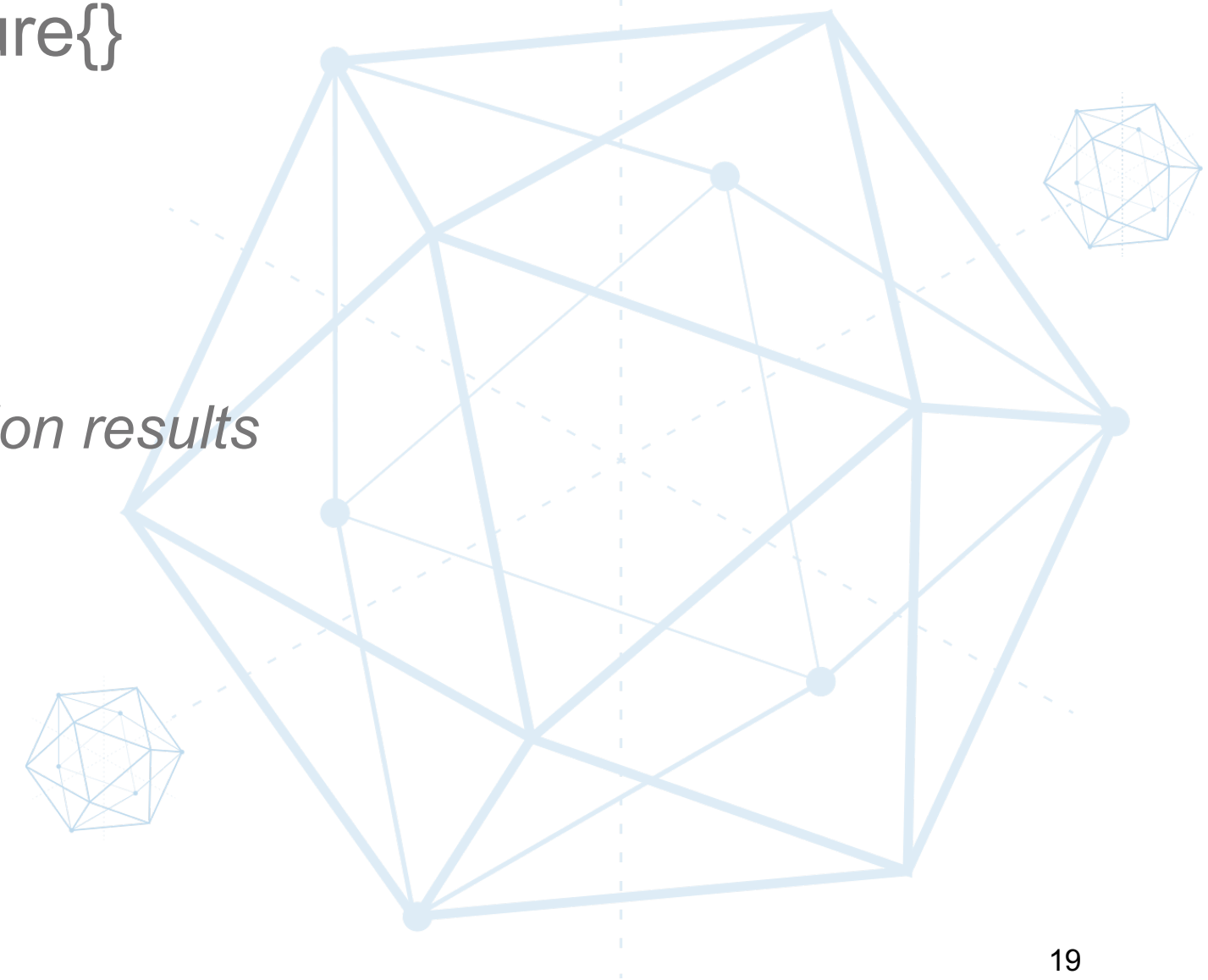
Configuration System ChainCode

- PeerConfiger{}
  - Handle those configuration transactions
- Init()
- Invoke()
  - JoinChain: *peer join into a chain*
  - UpdateConfigBlock: *update the configuration*
  - GetConfigBlock: *get the configuration block data*
  - GetChannels: *returns information about all channels for this peer*



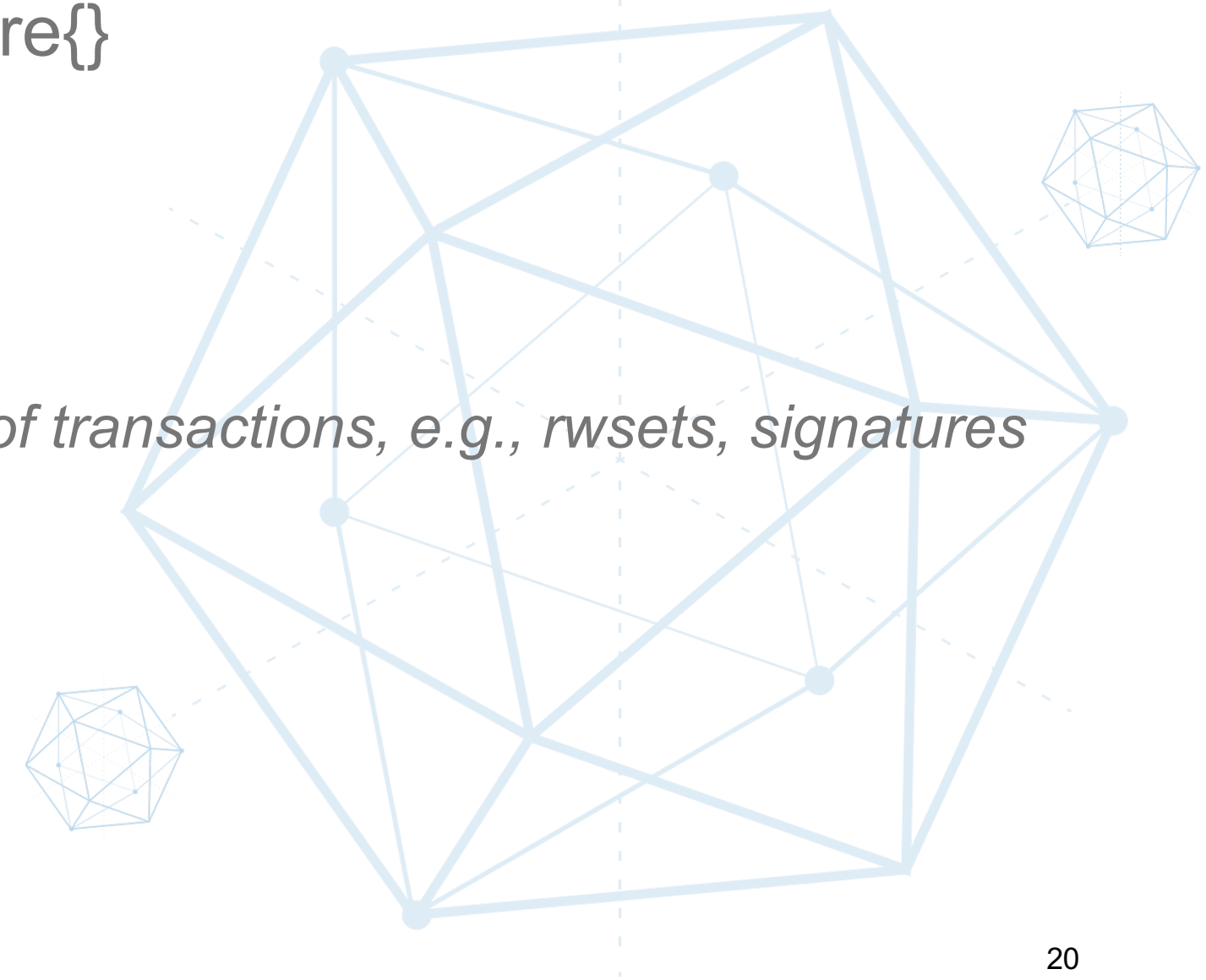
Endorsement System ChainCode

- EndorserOneValidSignature{}
  - Endorsement process
- Init()
- Invoke()
  - *Sign on chaincode's simulation results*
  - *More explicit rules (TBD)*



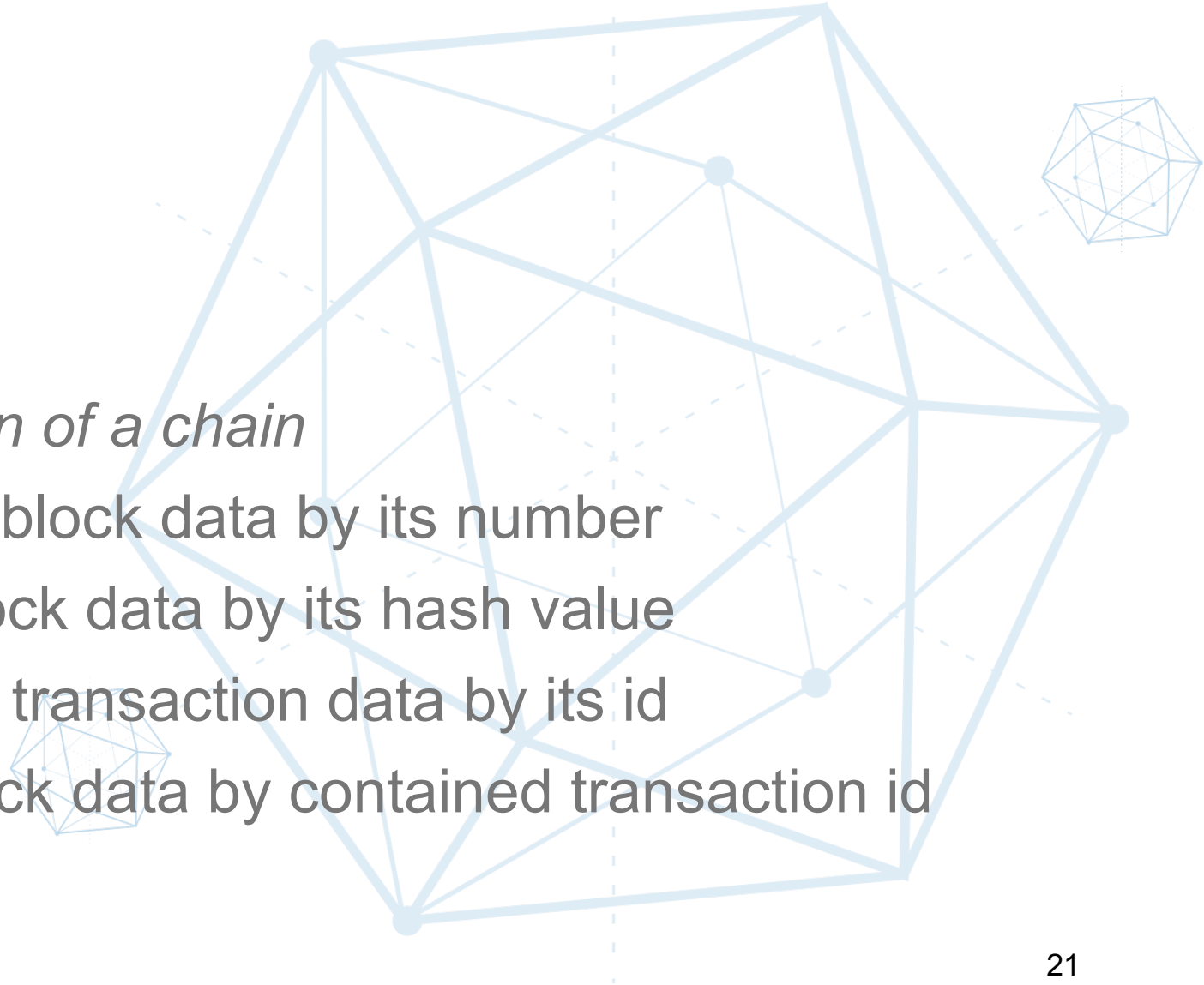
Validation System ChainCode

- ValidatorOneValidSignature{}
  - Validation process
- Init()
- Invoke()
  - *Validate the specified block of transactions, e.g., rwsets, signatures*



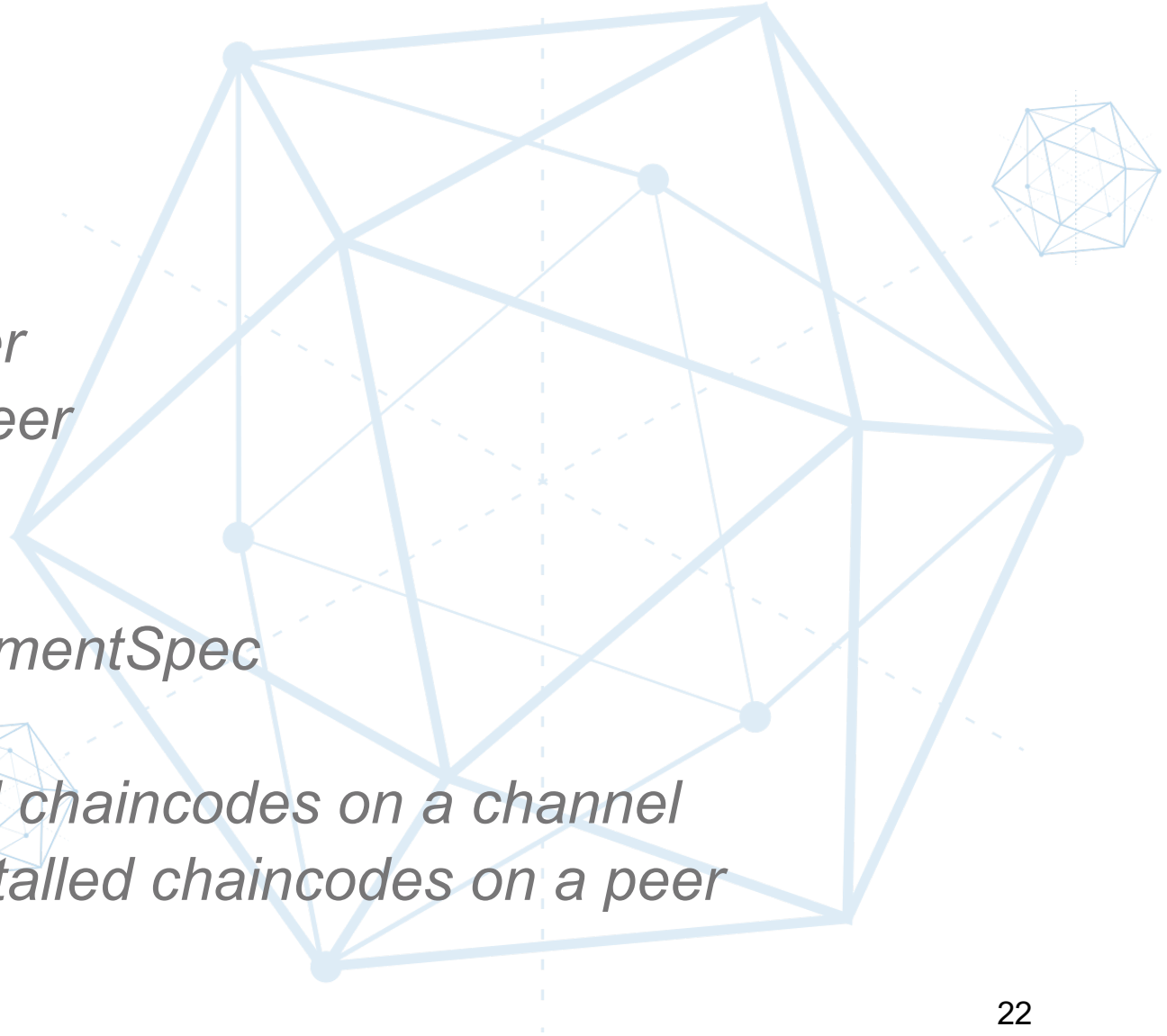
Query System ChainCode

- LedgerQuerier{}
  - Ledger query functions
- Init()
- Invoke()
  - *GetChainInfo: Get information of a chain*
  - GetBlockByNumber: Get the block data by its number
  - GetBlockByHash: Get the block data by its hash value
  - GetTransactionByID: Get the transaction data by its id
  - GetBlockByTxID: Get the block data by contained transaction id



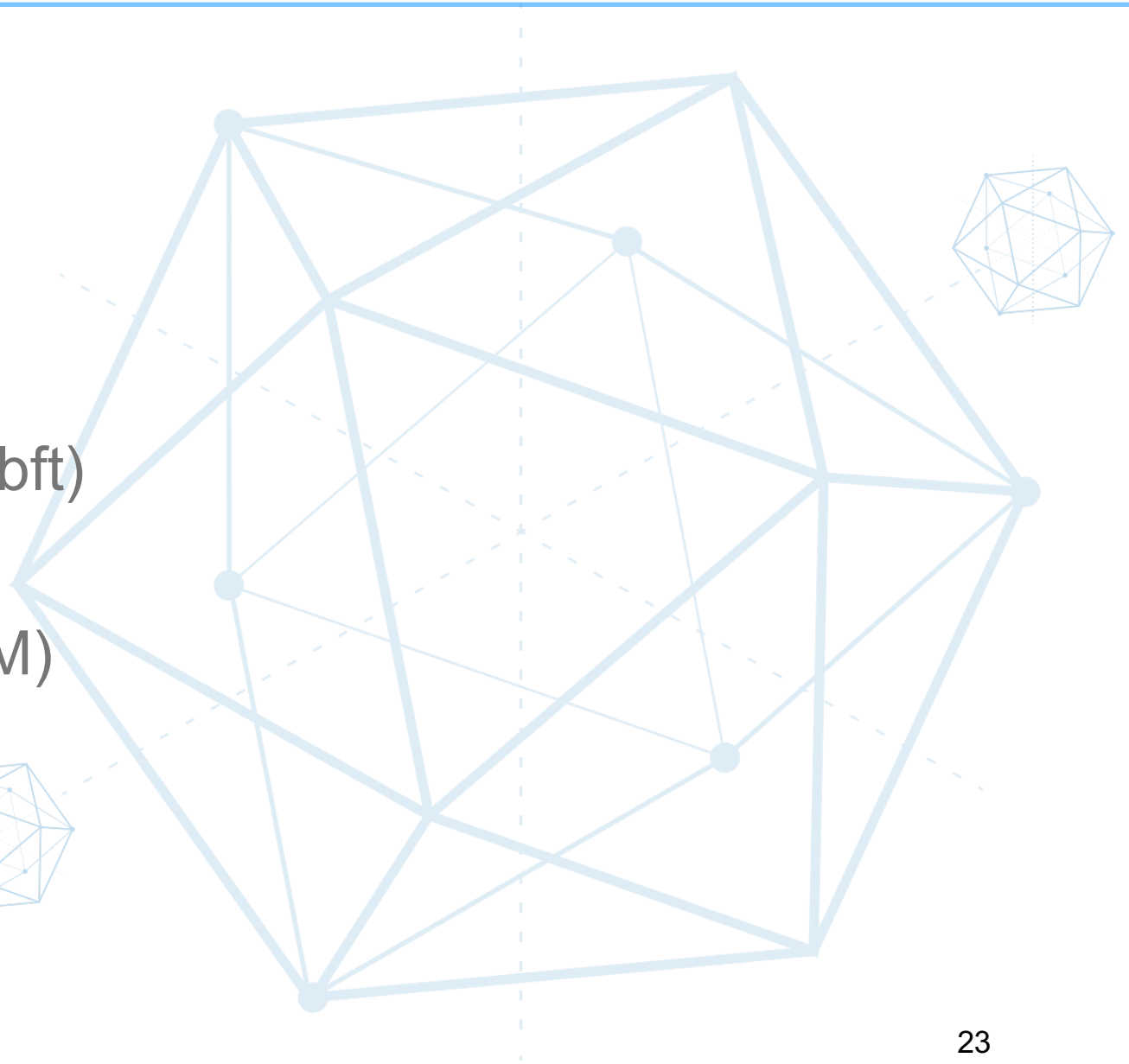
Life-cycle System ChainCode

- LifeCycleSysCC{}
  - Endorsement process
- Init()
- Invoke()
  - install: *install a chaincode on a peer*
  - deploy: *deploy a chaincode on a peer*
  - upgrade: *upgrade a chaincode*
  - getid: *get chaincode info*
  - getdepspec: *get ChaincodeDeploymentSpec*
  - getccdata: *get ChaincodeData*
  - getchaincodes: *get the instantiated chaincodes on a channel*
  - getinstalledchaincodes: *get the installed chaincodes on a peer*

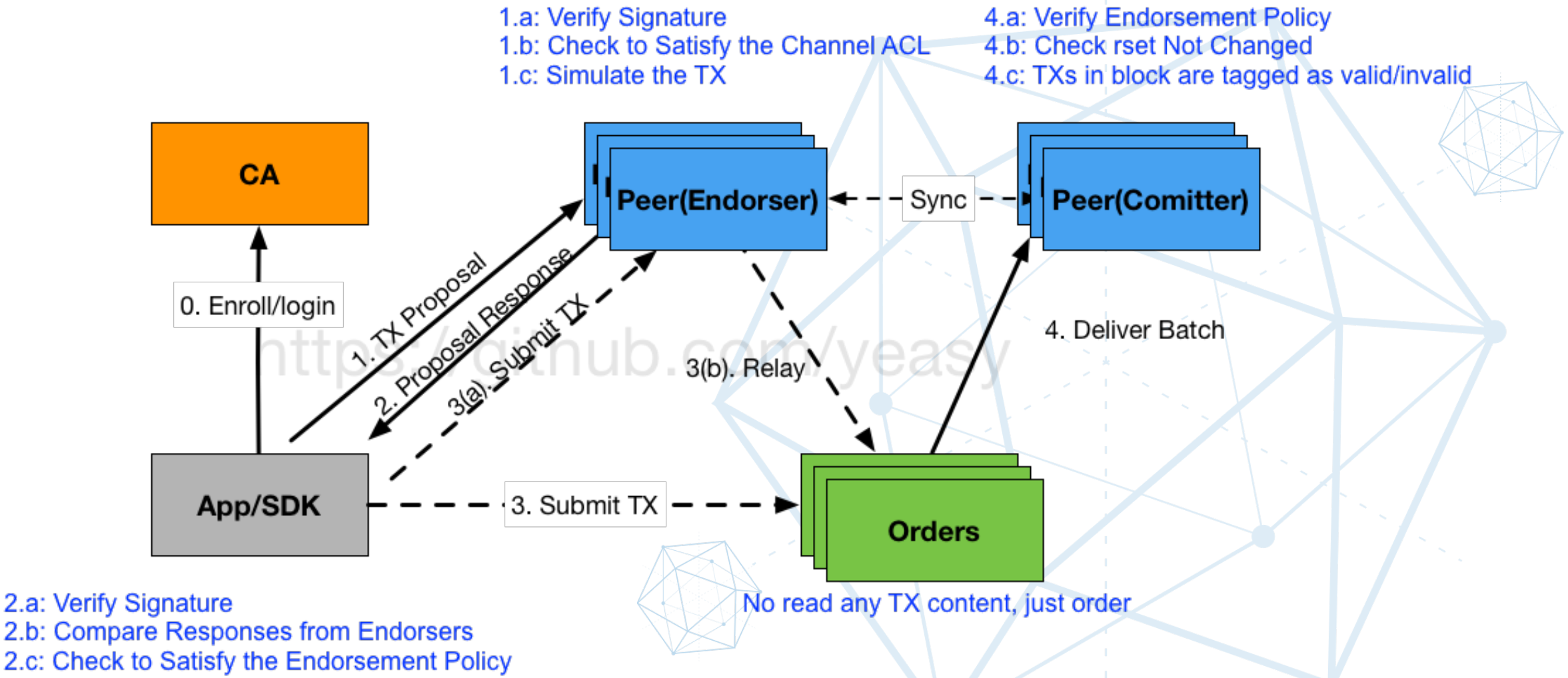


Pluggable Components

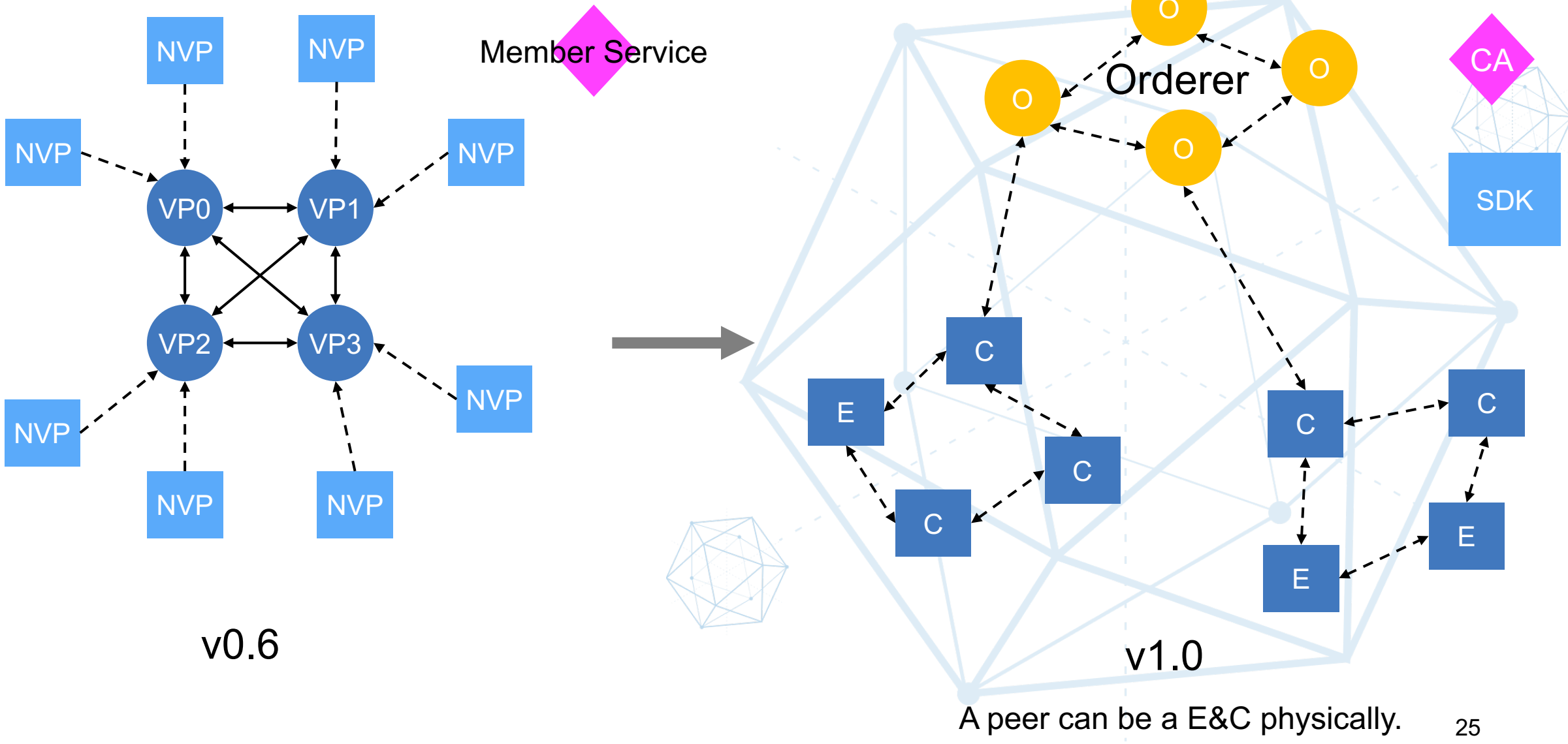
- Modular and Pluggable
  - Membership Services (CA)
  - SDKs (node, python, java, go)
  - Endorsement
  - Consensus service (solo, kafka, bft)
  - Ledger
  - Crypto algorithms (software, HSM)



Fabric 1.0 Workflow



Fabric 1.0 Deployment Scenarios



A peer can be a E&C physically.

Hyperledger Fabric Roadmap

Hack Fest docker images

- **60 participates tested**
- Basic v1 architecture in place
- Add / Remove Peers
- Channels
- Node SDK
- Go Chaincode
- Ordering Solo
- Fabric CA

V1 Alpha *

- Docker images
- Tooling to bootstrap network
- Fabric CA or bring your own
- Java and Node SDKs
- Ordering Services - Solo and Kafka
- Endorsement policy
- Level DB and Couch DB
- Block dissemination across peers via Gossip

V1 GA *

- Hardening, usability, serviceability, load, operability and stress test
- Java Chaincode
- Chaincode ACL
- Chaincode packaging & LCI
- Pluggable crypto
- HSM support
- Consumability of configuration
- Next gen bootstrap tool (config update)
- Config transaction lifecycle
- Eventing security
- Cross Channel Query
- Peer management APIs
- Documentation

V Next *

- SBFT
- Archive and pruning
- System Chaincode extensions
- Side DB for private data
- Application crypto library
- Dynamic service discovery
- REST wrapper
- Python SDK
- Identity Mixer (Stretch)
- Tcerts

2016/17 December | March | June | Future

Connect-a-thon

- 11 companies in Australia, Hungary, UK, US East Coast, US West Coast, Canada dynamically added peers and traded assets

Connect-a-cloud

- Dynamically connecting OEM hosted cloud environments to trade assets

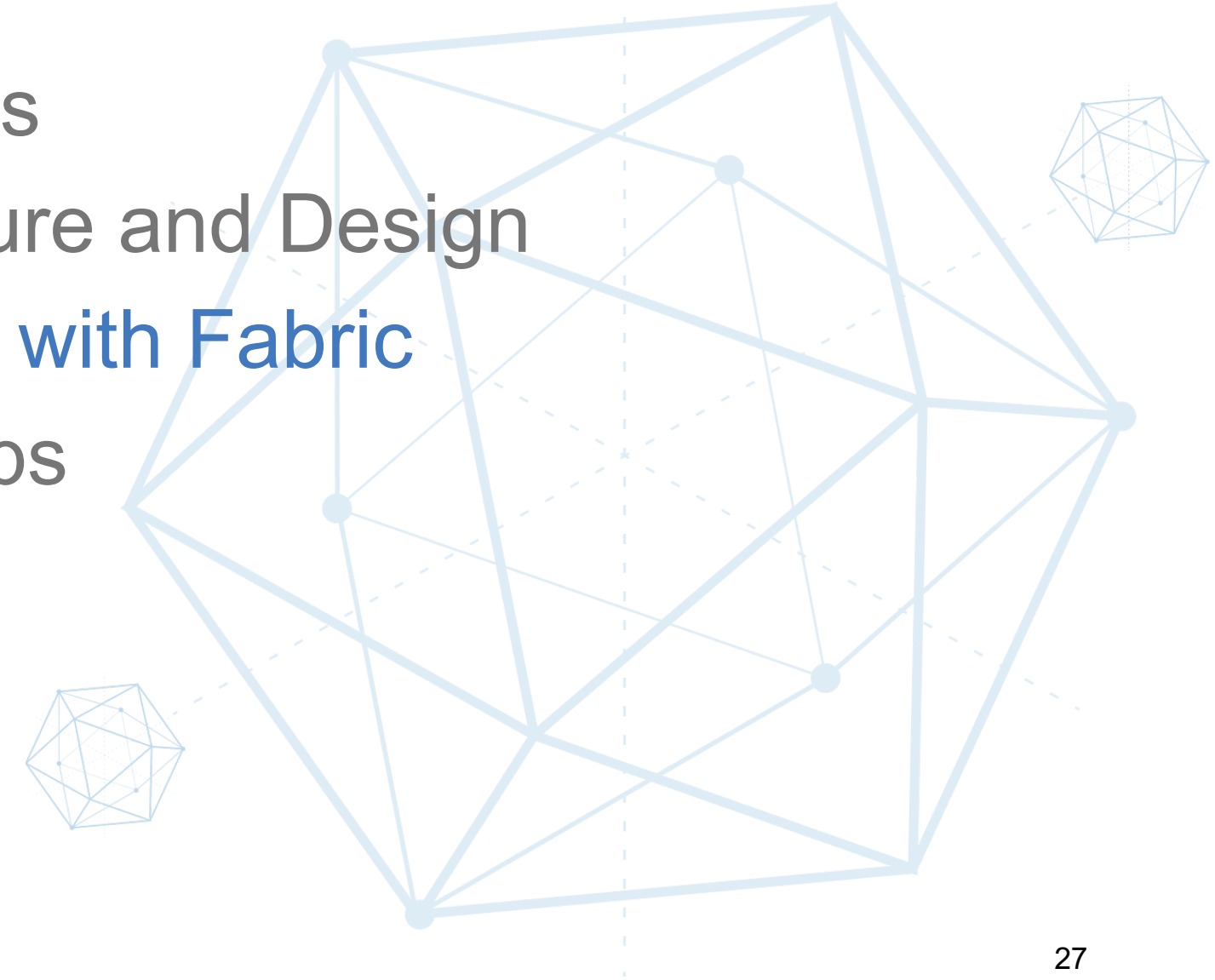
HYPERLEDGER
BLOCKCHAIN TECHNOLOGIES FOR BUSINESS

*** Dates for Alpha, Beta, and GA are determined by Hyperledger community and are currently proposals.**

Proposed Alpha detailed content:
https://wiki.hyperledger.org/projects/proposedv1alphacontent

Outline

- Hyperledger Projects
- Fabric 1.0 Architecture and Design
- Develop Application with Fabric
- Deploy and Run Apps
- Q&A

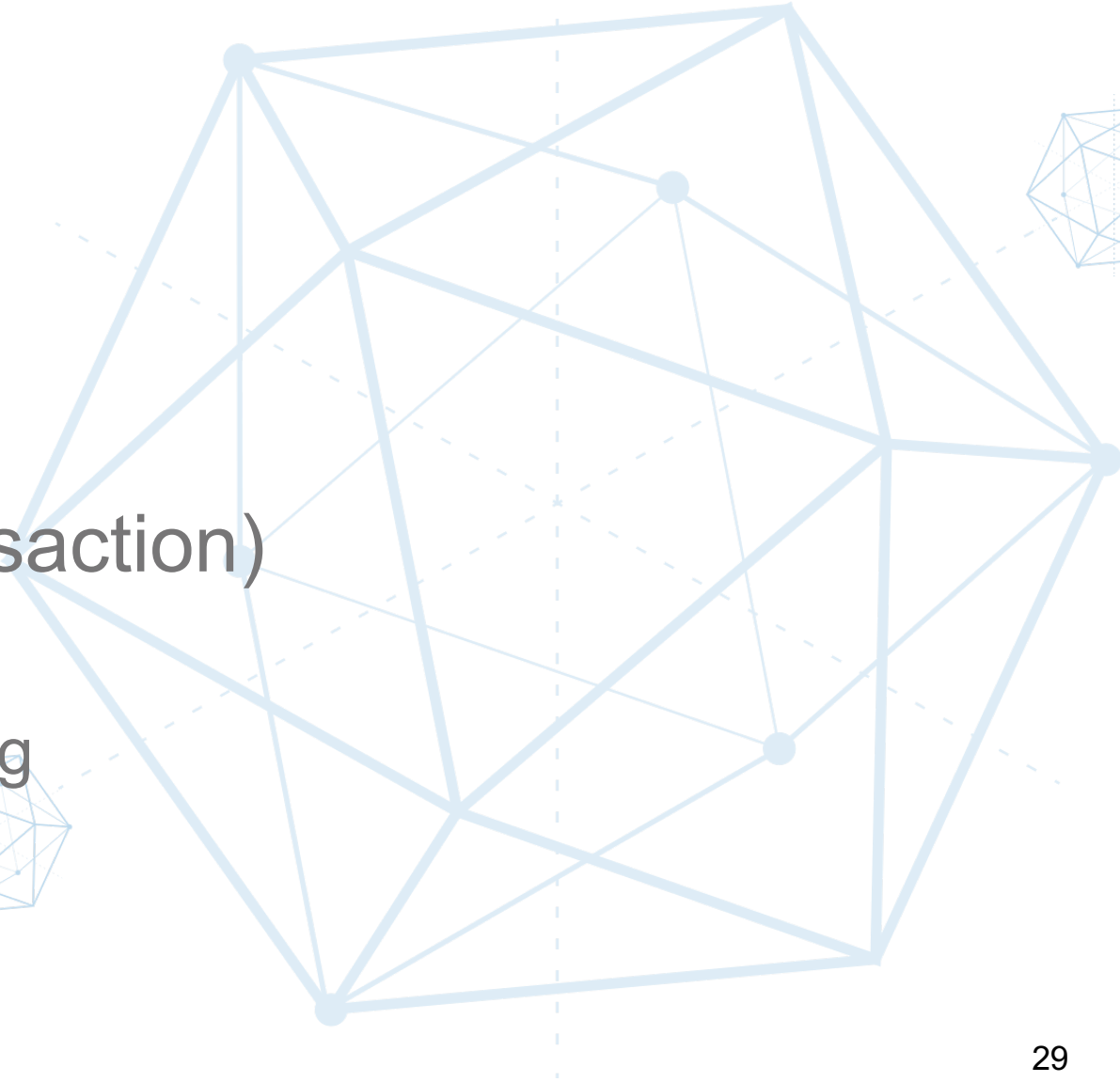


Chaincode

- Application Logics to Interact with the Ledger
  - E.g., transfer money from account A to B
- Implement a Chaincode Interface
  - fabric/core/chaincode/shim/interface.go
- Running in Containers
- Call Peer's API by a ChaincodeStubInterface
  - fabric/core/chaincode/shim/interface.go
- Stateless and Deterministic

Chaincode Support

- Languages
  - Golang
  - Java
  - More to Support
- How to Call Chaincode (Transaction)
  - CLI
  - SDK: Node, Python, Java, Golang



Chaincode Programming

- Implement the Chaincode interface

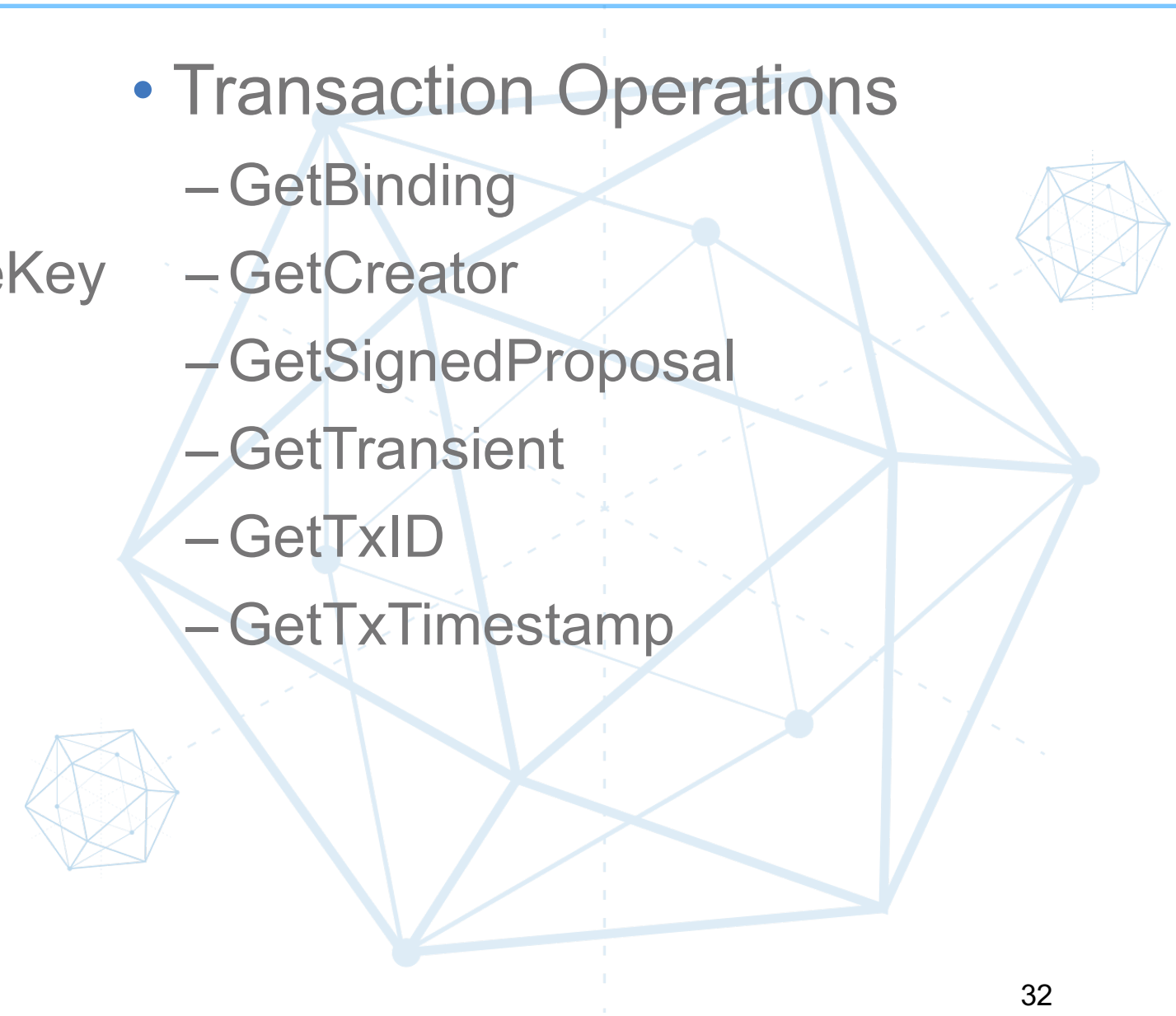
```
type Chaincode interface {
   // Init is called during Deploy transaction after the container has been
   // established, allowing the chaincode to initialize its internal data
   Init(stub ChaincodeStubInterface) pb.Response
   // Invoke is called for every Invoke transactions. The chaincode may change
   // its state variables
   Invoke(stub ChaincodeStubInterface) pb.Response
}
```

Chaincode Programming

```
package main
import (
        "errors"
        "fmt"
        "github.com/hyperledger/fabric/core/chaincode/shim"
)
type DemoChaincode struct { }
func (t *DemoChaincode) Init(stub shim.ChaincodeStubInterface) pb.Response {
        // more logics using stub here
        return stub.Success(nil)
}

func (t *DemoChaincode) Invoke(stub shim.ChaincodeStubInterface) pb.Response
        // more logics using stub here
        return stub.Success(nil)

}

func main() {
        err := shim.Start(new(DemoChaincode))
        if err != nil {
                fmt.Printf("Error starting DemoChaincode: %s", err)
        }
}
```

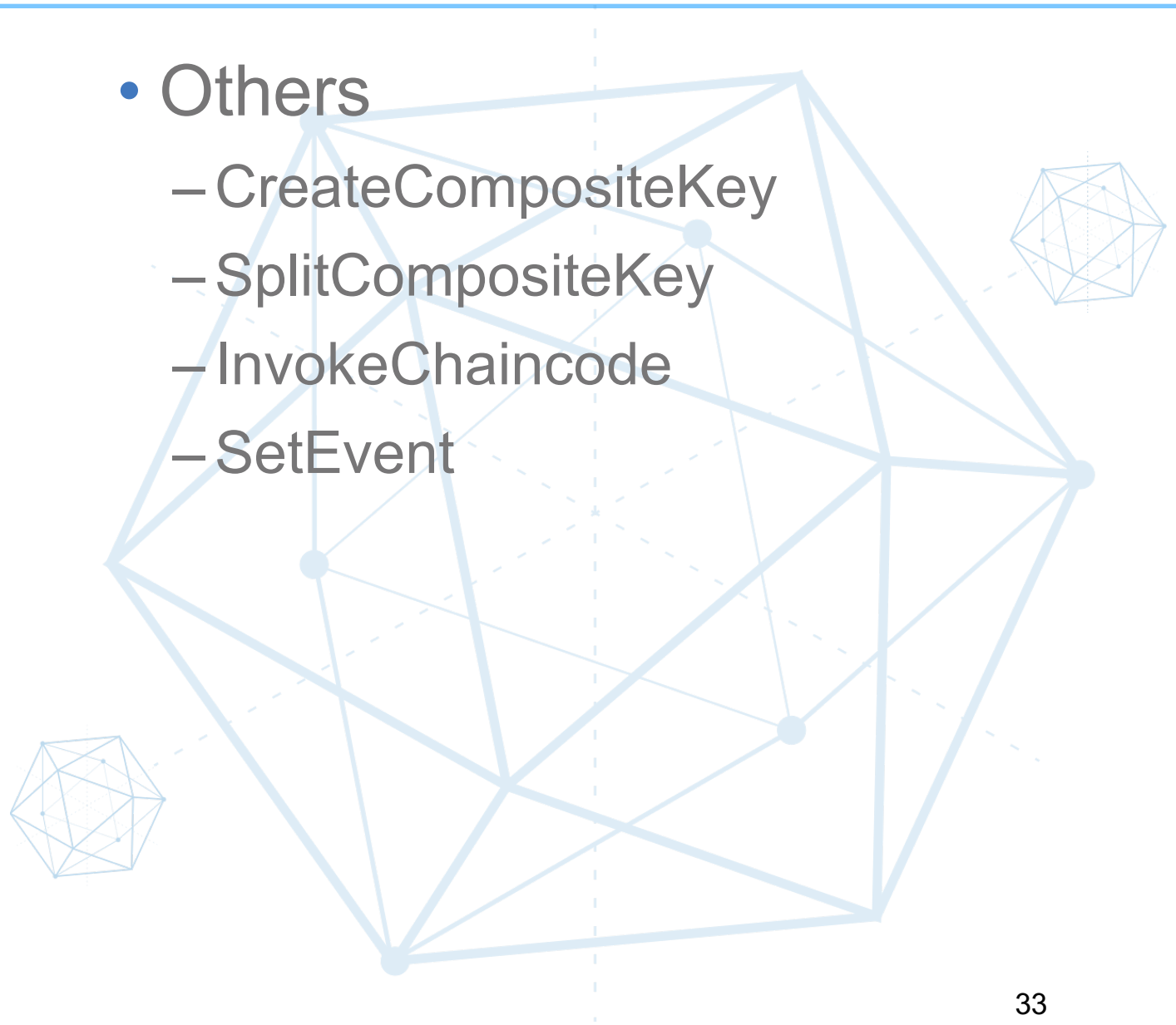
ChaincodeStubInterface

- Ledger State Operations
  - GetState
  - GetStateByPartialCompositeKey
  - GetStateByRange
  - DelState
  - PutState
  - GetHistoryForKey
  - GetQueryResult
- Transaction Operations
  - GetBinding
  - GetCreator
  - GetSignedProposal
  - GetTransient
  - GetTxID
  - GetTxTimestamp



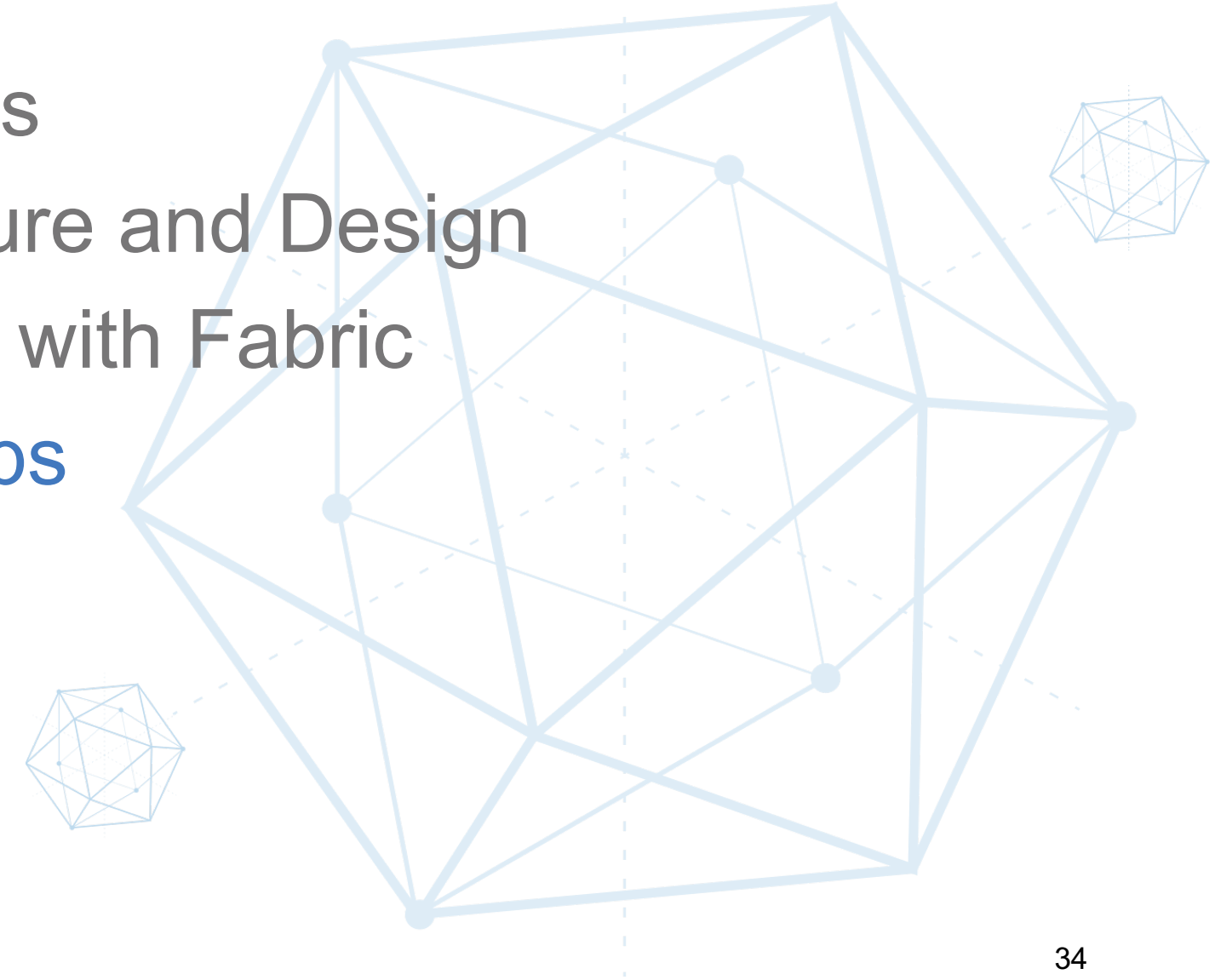
ChaincodeStubInterface

- Stub Arguments
  - GetArgs
  - GetArgsSlice
  - GetFunctionAndParameters
  - GetStringArgs
- Others
  - CreateCompositeKey
  - SplitCompositeKey
  - InvokeChaincode
  - SetEvent



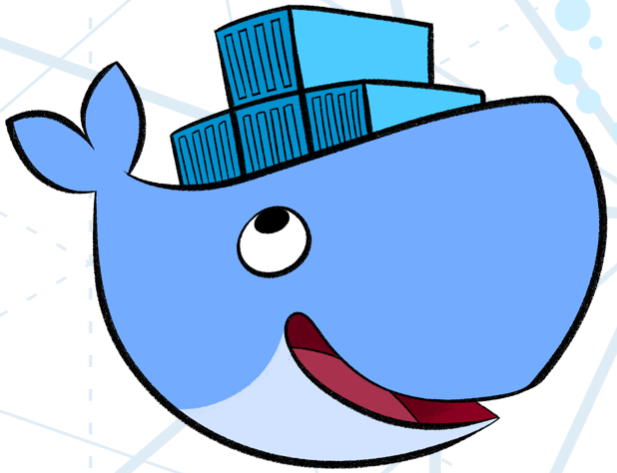
Outline

- Hyperledger Projects
- Fabric 1.0 Architecture and Design
- Develop Application with Fabric
- Deploy and Run Apps
- Q&A



Environment Setup – Docker Installation

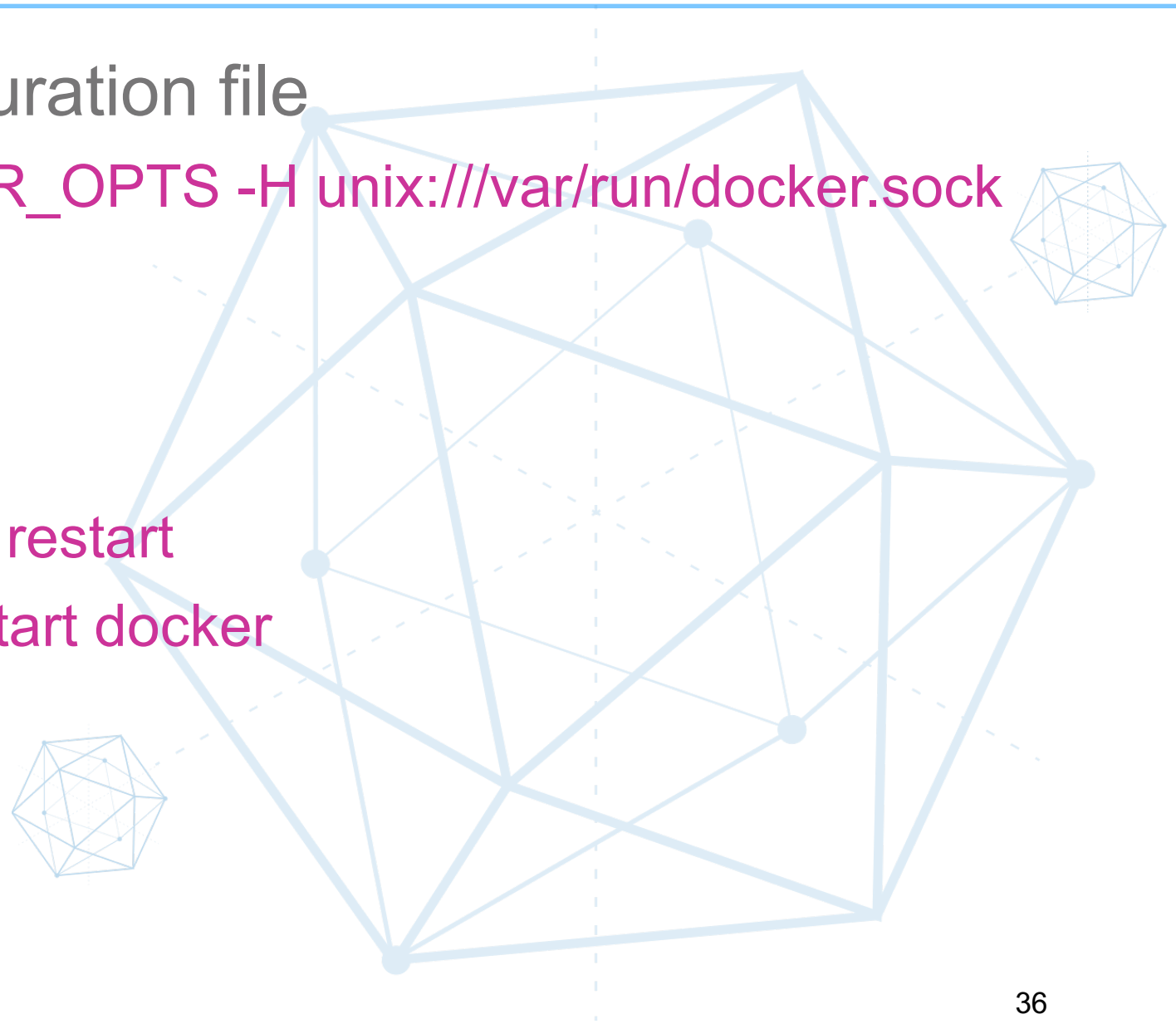
- Docker 1.12+
- Linux
  - 64 bit
  - kernel 3.10+
  - curl -sSL https://get.docker.com/ | sh
- Mac
  - Docker for Mac
- Docker-Compose 1.7.0+
  - pip install docker-compose>=1.7.0



* Non-container deployments are supported.

Environment Setup - Configuration

- Update the Docker configuration file
  - DOCKER_OPTS="$DOCKER_OPTS -H unix:///var/run/docker.sock -H tcp://0.0.0.0:2375"
- Restart Docker Daemon
  - Upstart: sudo service docker restart
  - Systemd: sudo systemctl restart docker

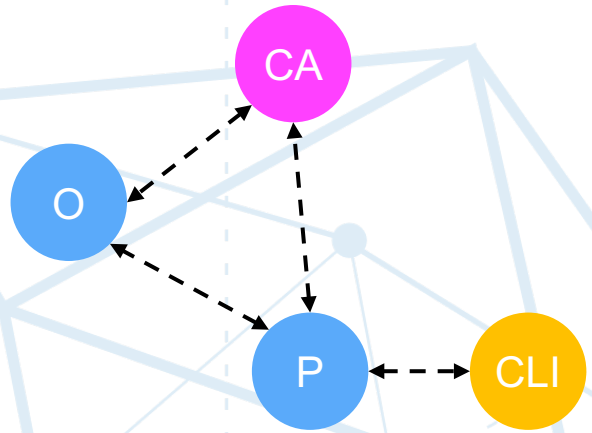


Fabric 1.0 Bootup in 3 steps

- Get Docker images
  - https://github.com/yeasy/docker-compose-files/tree/master/hyperledger/1.0
  - http://ibm.com/ibm/cn/blockchain/
  - https://hub.docker.com/r/hyperledger
- Get Compose file
  - git clone https://github.com/yeasy/docker-compose-files
- Start fabric
  - cd hyperledger/1.0 & docker-compose up

Play Transactions

- Check container status
  - watch docker ps
- Enter the cli container
  - docker exec -it fabric-cli bash




```
$ docker ps
CONTAINER ID        IMAGE                          COMMAND
c1cf099e1f76        hyperledger/fabric-peer        "bash -c 'while tr..."
0b67c42fd5cc        hyperledger/fabric-peer        "peer node start -..."
80b5fb85636e        hyperledger/fabric-orderer     "orderer"
f3680e5889b0        hyperledger/fabric-ca          "fabric-ca-server ..."
```

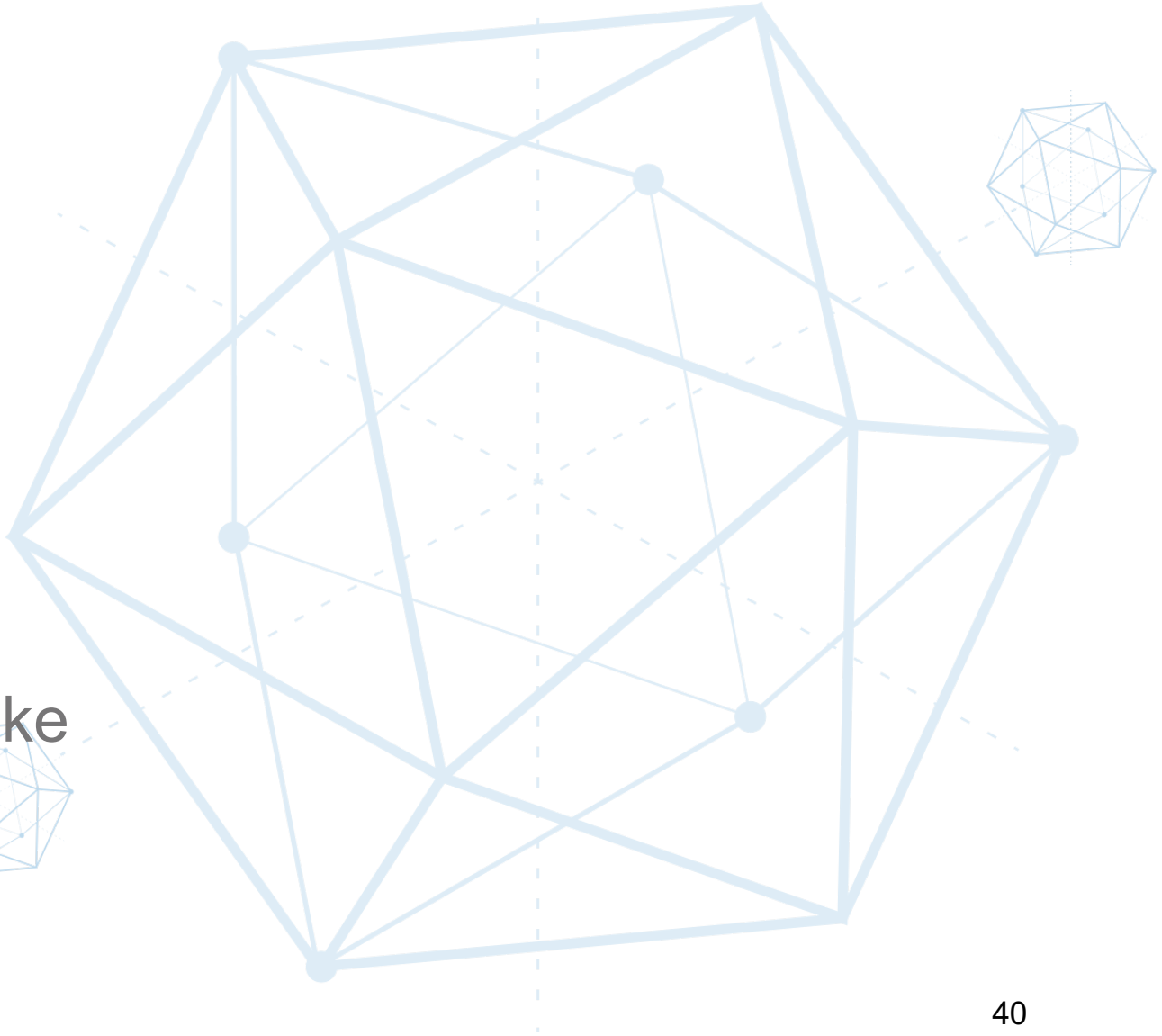
Play Transactions cont.

- Install/instantiate chaincode
  - CC_PATH= github.com/hyperledger/fabric/examples/chaincode/go/chaincode_example02
  - peer chaincode install -v 1.0 -n test_cc -p $CC_PATH -c '{"Args":["init","a","100","b","200"]}'
  - peer chaincode instantiate -v 1.0 -n test_cc -p $CC_PATH -c '{"Args":["init","a","100","b","200"]}'
- Invoke chaincode
  - peer chaincode invoke -n test_cc -c '{"Args":["query","a"]}'
  - peer chaincode invoke -n test_cc -c '{"Args":["invoke","a","b","10"]}'

```
$ docker ps
CONTAINER ID        IMAGE                               COMMAND
c0abb4b9206b        dev-peer0-test_cc-1.0               "chaincode -peer.a..."
c1cf099e1f76        hyperledger/fabric-peer             "bash -c 'while tr..."
0b67c42fd5cc        hyperledger/fabric-peer             "peer node start -..."
80b5fb85636e        hyperledger/fabric-orderer          "orderer"
f3680e5889b0        hyperledger/fabric-ca               "fabric-ca-server ..."
```

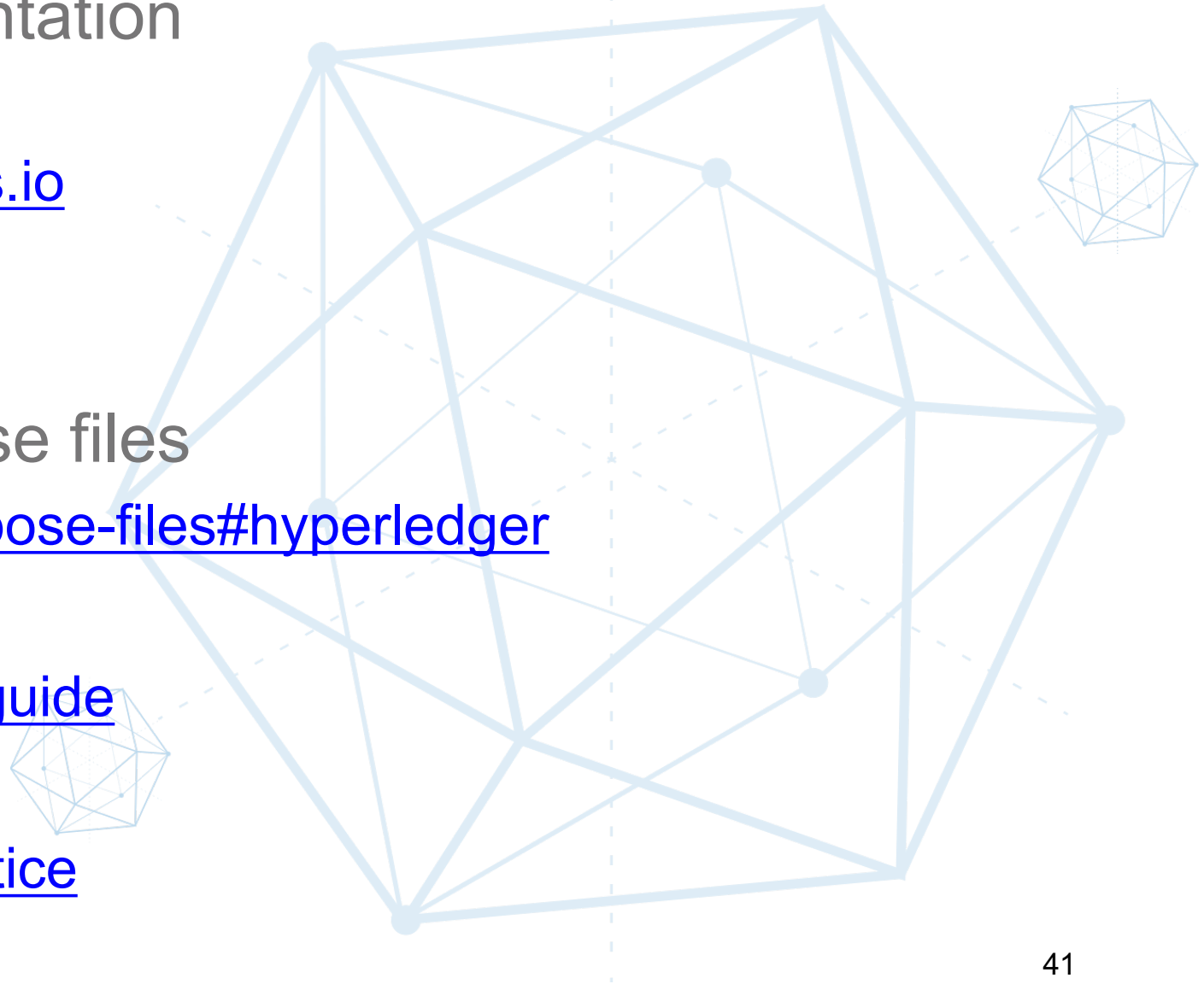
More on Using Fabric

- Application interactions
  - APIs: gRPC
  - SDK: Node, Python, Java
- Commands
  - Peer start/stop
  - Channel create/join
  - User enroll/login
  - Chaincode install/instantiate/invoke



Reference

- Hyperledger Wiki&Documentation
  - wiki.hyperledger.org
  - hyperledger-fabric.readthedocs.io
- IBM 区块链
  - ibm.com/ibm/cn/blockchain/
- Hyperledger Fabric Compose files
  - github.com/yeasy/docker-compose-files#hyperledger
- 《区块链技术指南》
  - github.com/yeasy/blockchain_guide
- 《Docker 从入门到实践》
  - github.com/yeasy/docker_practice



Questions?

Thank You!
@baohua

Slides available at github.com/yeasy/seminar-talk