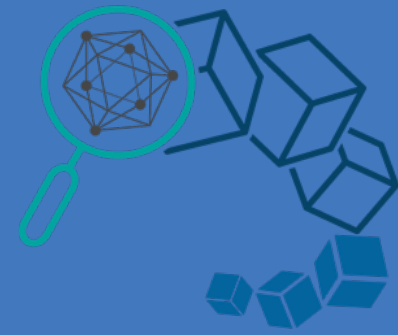




**HYPERLEDGER**  
BLOCKCHAIN TECHNOLOGIES FOR BUSINESS



# Hyperledger Fabric Chaincode Development

Baohua Yang  
Dec, 2017

# About Me

- **Interested Areas**

- Fintech, Cloud and Analytics

- **Technical Leader**

- Senior Researcher/Architect in IBM, Oracle

- **Open-Source Contributor**

- [Hyperledger](#), [OpenStack](#), [OpenDaylight](#), etc.

- **Hyperledger Developer**

- Core designer & committer of [Fabric](#), [Cello](#), [sdk](#) etc.

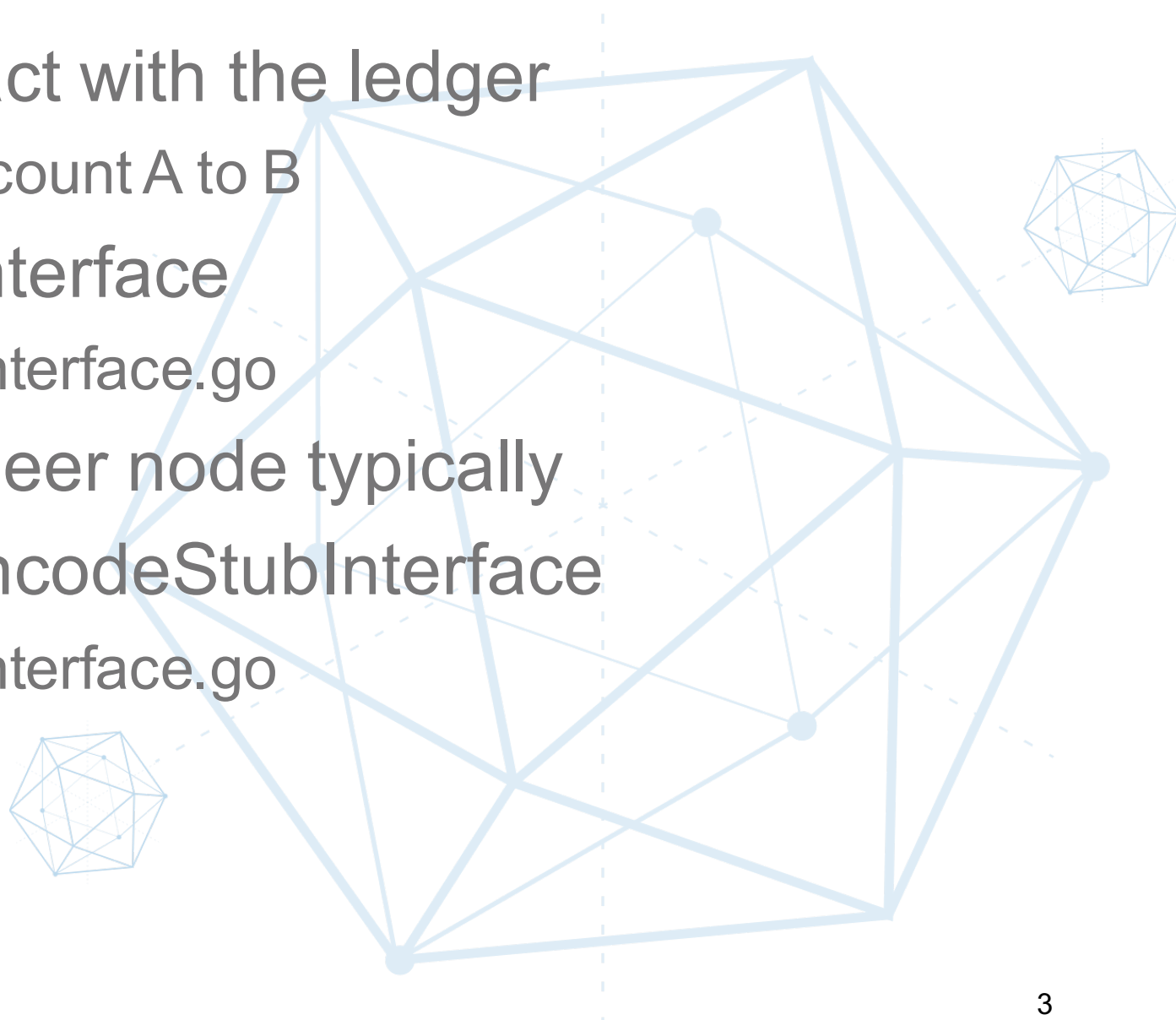
- [Hyperledger Technical Steering Committee \(TSC\)](#) Member

- [Hyperledger Technical Working Group China](#) Chair



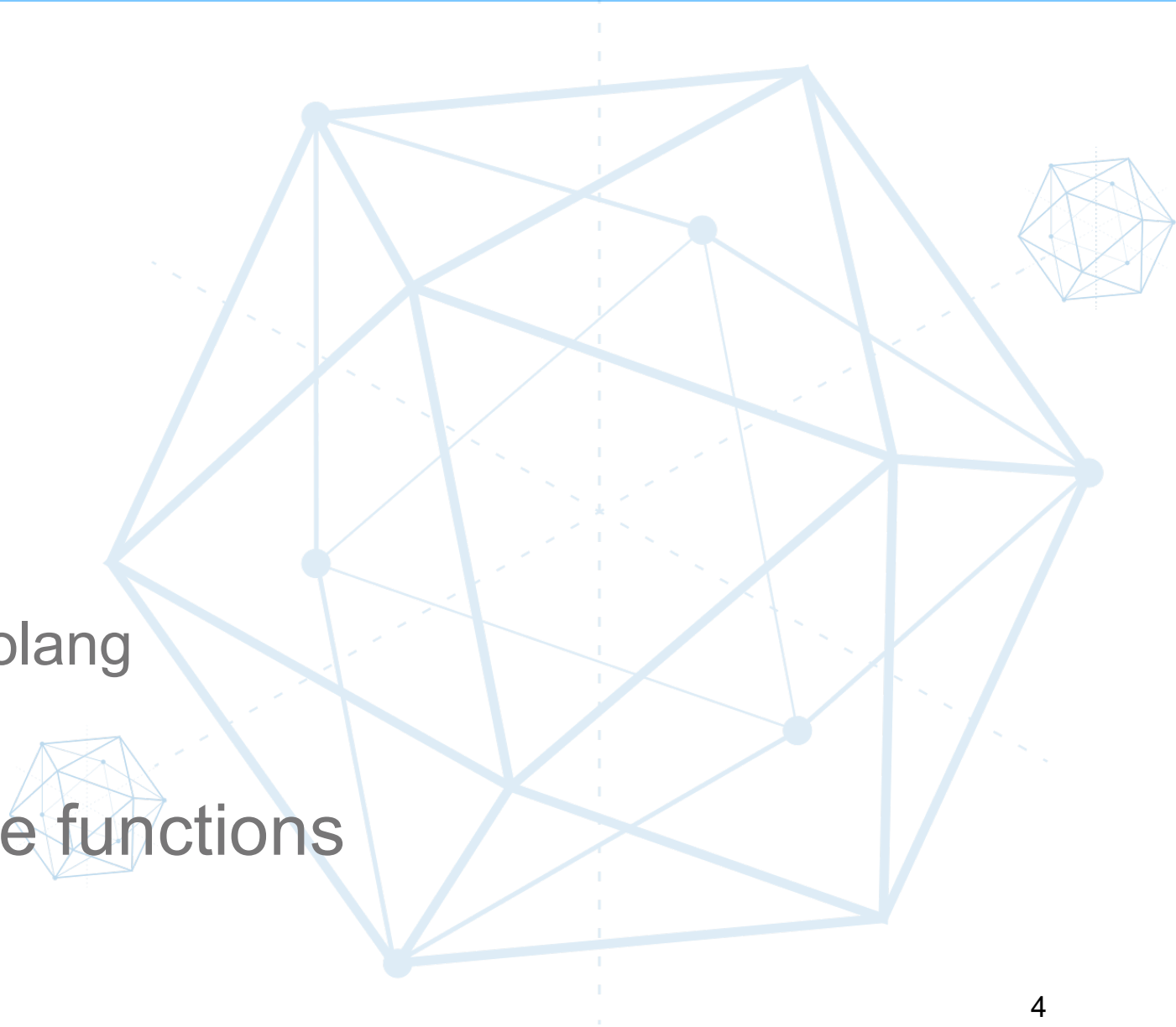
# Chaincode Concept

- 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 at peer node typically
- Call Peer's API by a ChaincodeStubInterface
  - `fabric/core/chaincode/shim/interface.go`
- Stateless
- Deterministic



# Chaincode Support

- Languages
  - Golang
  - Java
- How to call chaincode
  - CLI
  - SDK: Node, Python, Java, Golang
- Transaction: call chaincode functions



# 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
```

```
}
```

# Demo Chaincode

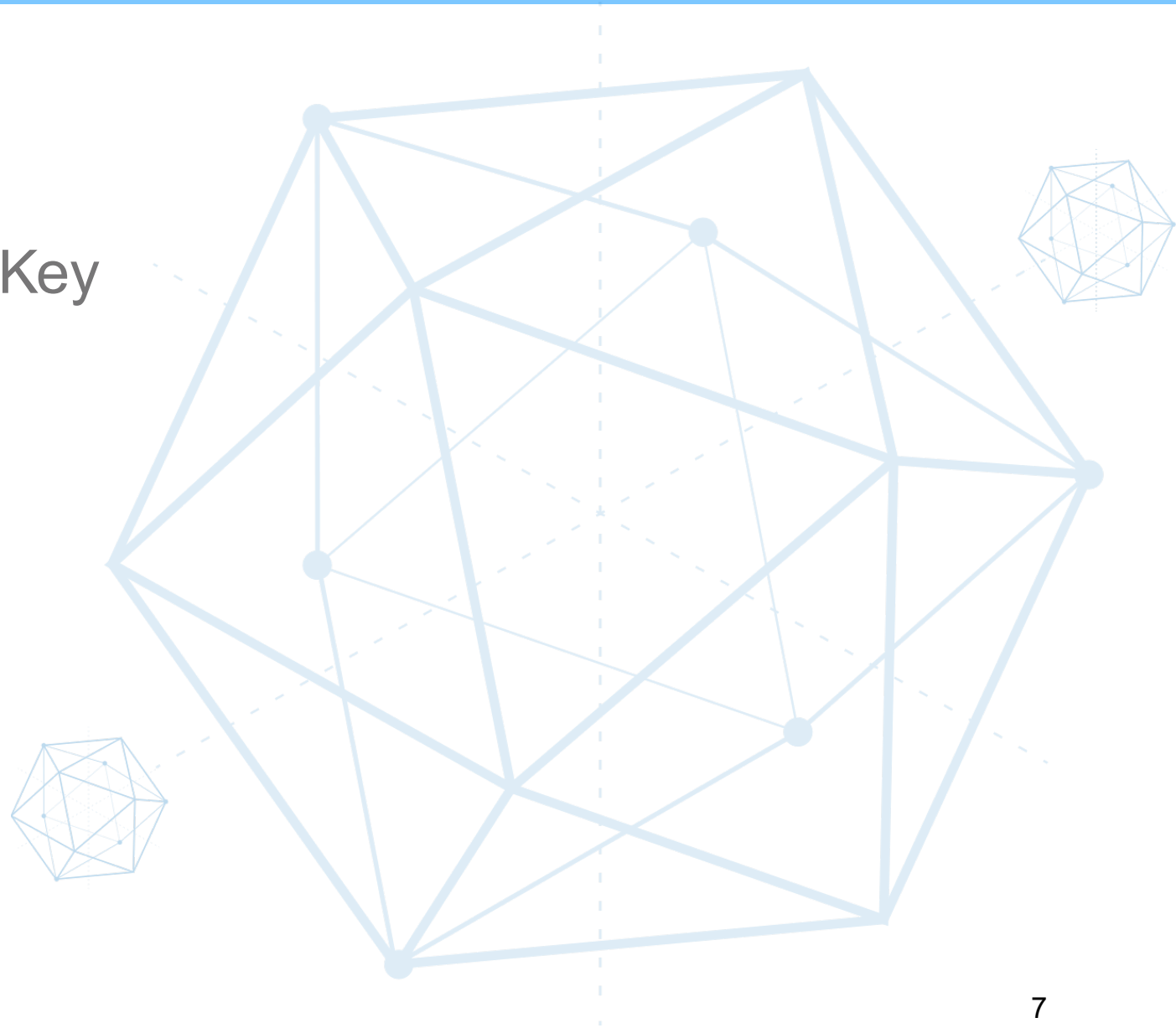
```
package main
import (
    "errors"
    "fmt"
    "github.com/hyperledger/fabric/core/chaincode/shim"
)
type DemoChaincode struct { }
func (t *DemoChaincode) Init(stub shim.ChaincodeStubInterface) pb.Response {
    // more logics using stub here
    return stub.Success(nil)
}

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)
    }
}
```

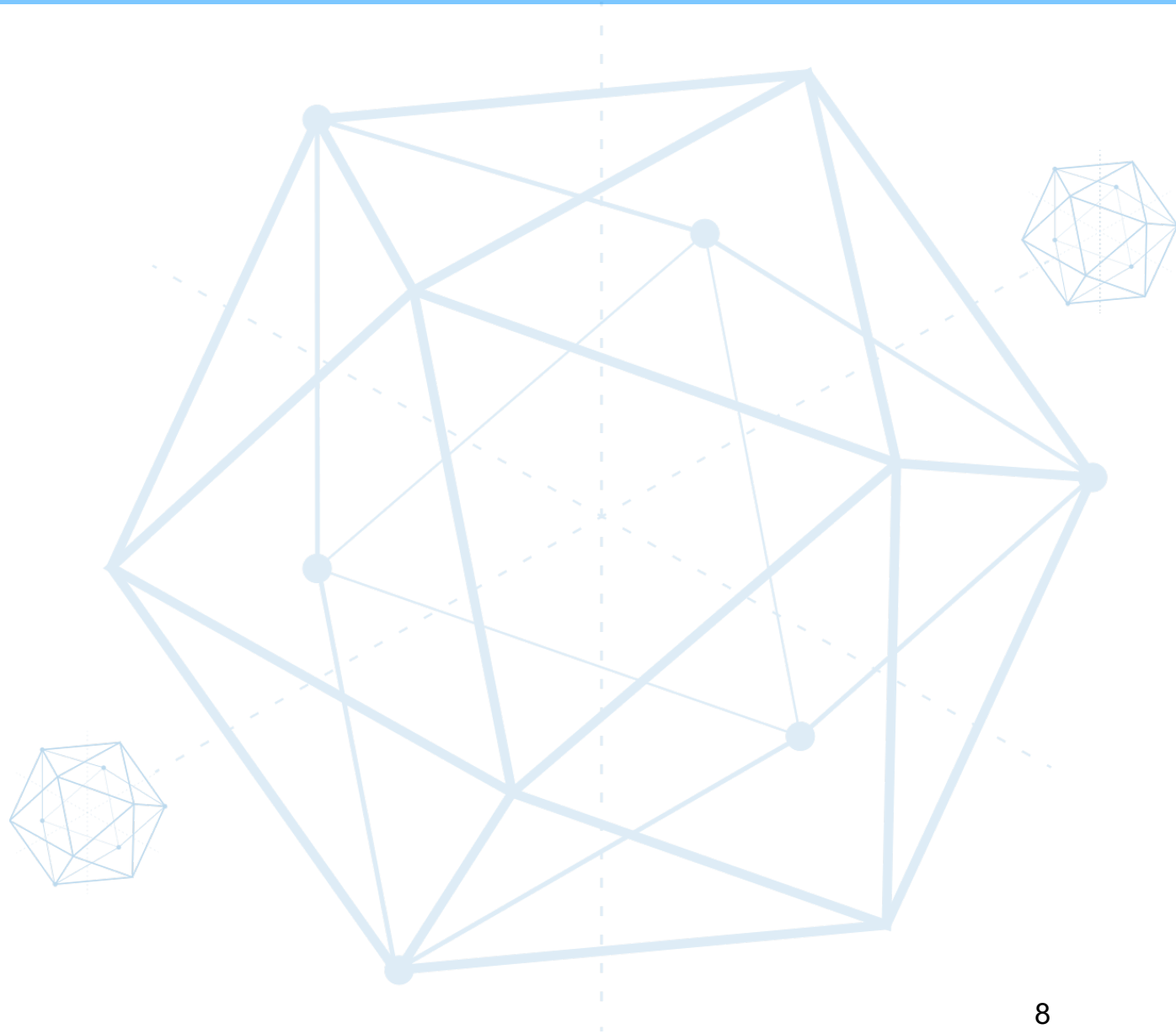
# ChaincodeStubInterface

- Ledger State Operations
  - GetState
  - GetStateByPartialCompositeKey
  - GetStateByRange
  - DelState
  - PutState
  - GetHistoryForKey
  - GetQueryResult



# ChaincodeStubInterface

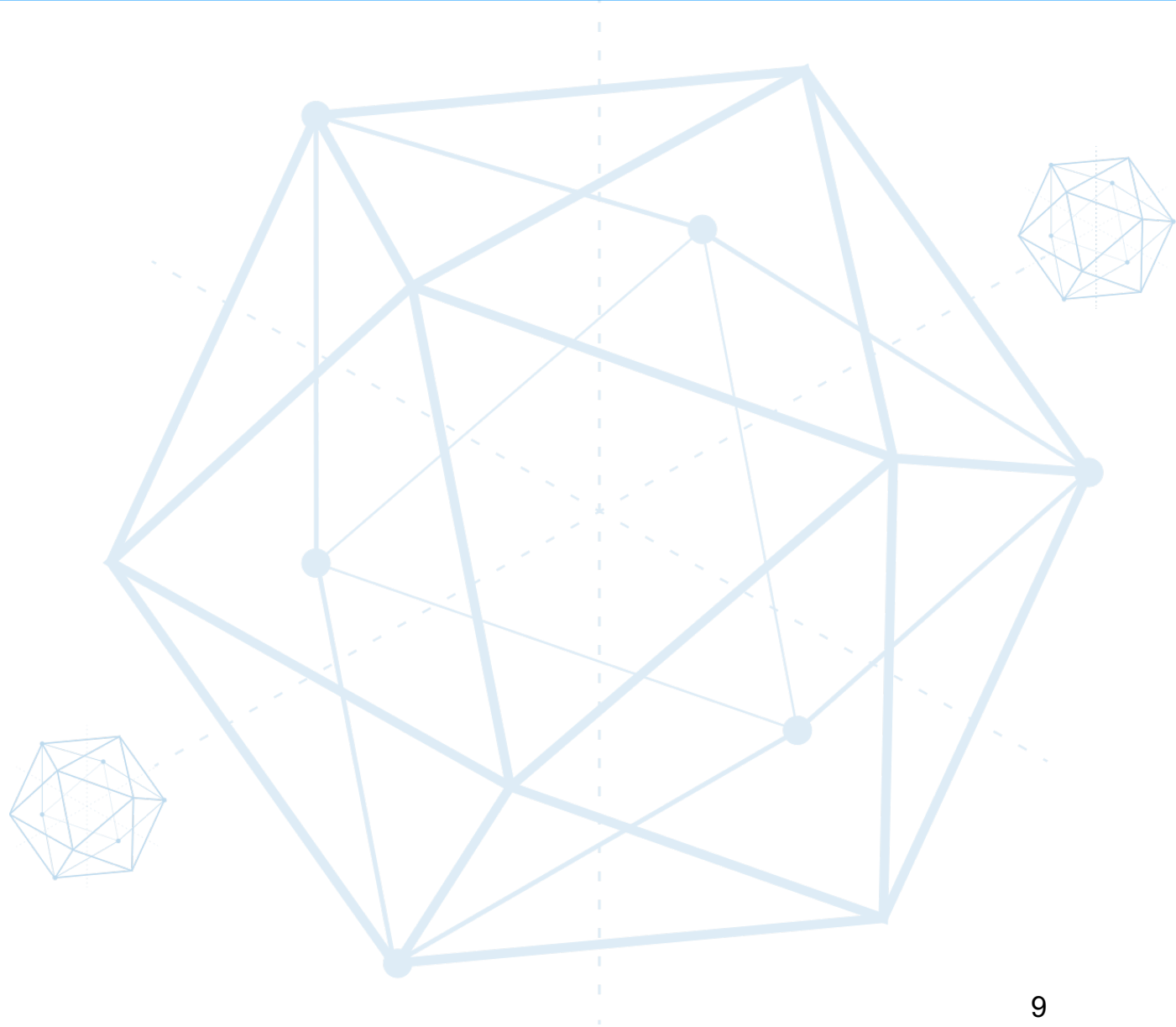
- Transaction Operations
  - GetBinding
  - GetCreator
  - GetSignedProposal
  - GetTransient
  - GetTxID
  - GetTxTimestamp





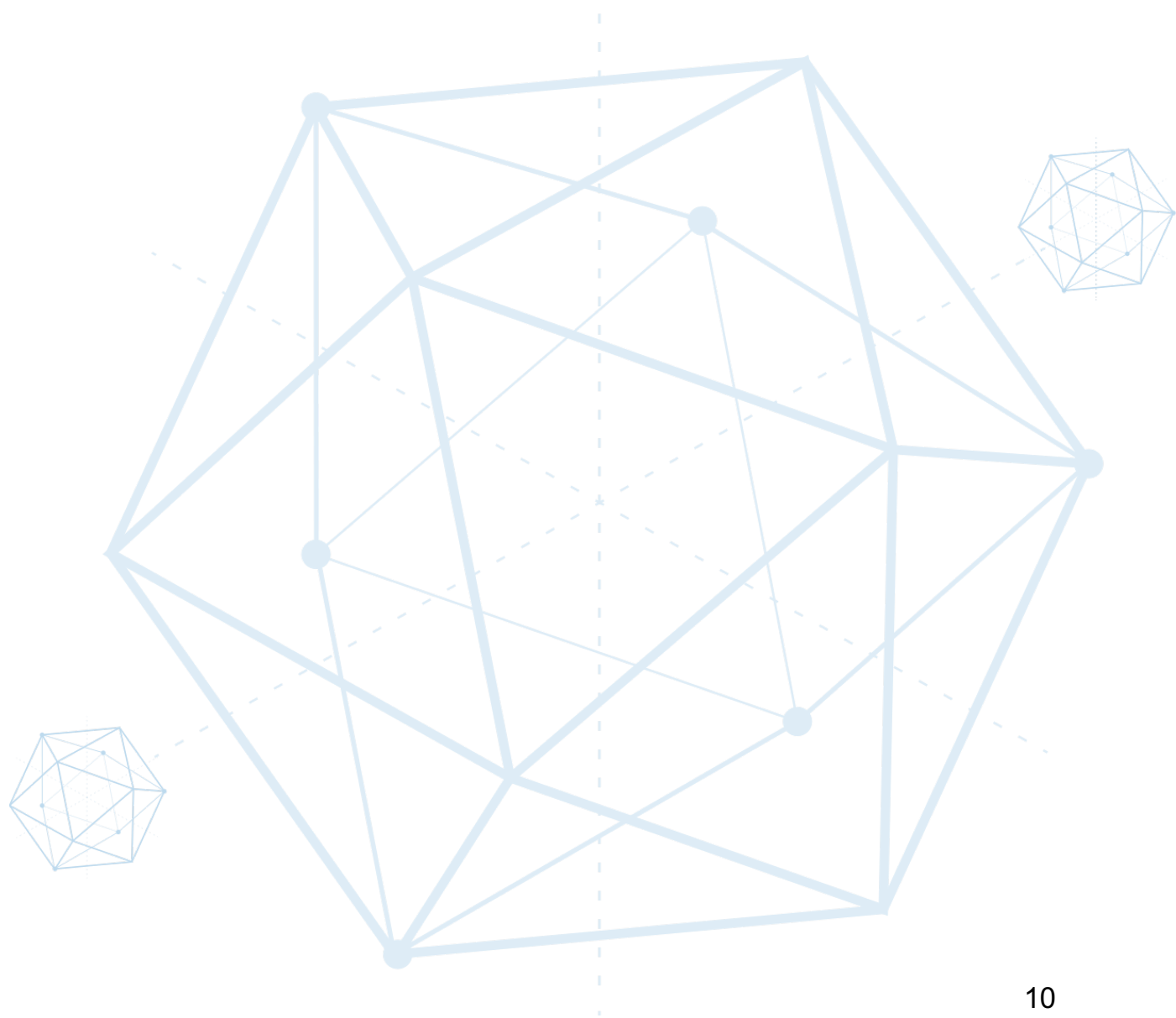
# ChaincodeStubInterface

- Stub Arguments
  - GetArgs
  - GetArgsSlice
  - GetFunctionAndParameters
  - GetStringArgs



# ChaincodeStubInterface

- Others
  - CreateCompositeKey
  - SplitCompositeKey
  - InvokeChaincode
  - SetEvent



# Reference

- [Hyperledger Project](#)
- 《[区块链原理设计与应用](#)》
- 《[Docker 技术入门与实战](#)》
- [超级账本 Fabric 源码剖析](#)
- [github.com/yeasy/blockchain\\_guide](https://github.com/yeasy/blockchain_guide)





**HYPERLEDGER**  
BLOCKCHAIN TECHNOLOGIES FOR BUSINESS



# Questions?

**Thank You!**  
**@baohua**

*Slides available at [github.com/yeasy/seminar-talk#hyperledger](https://github.com/yeasy/seminar-talk#hyperledger)*