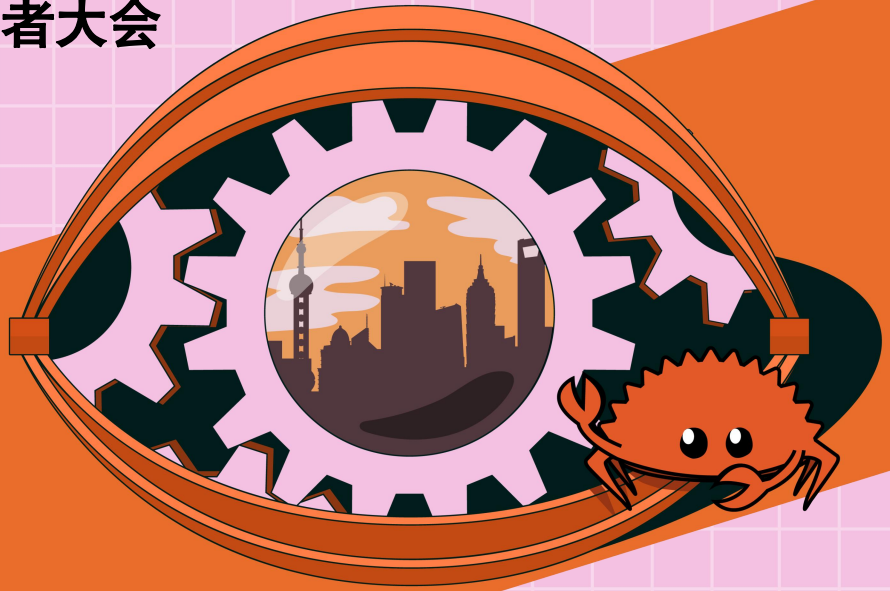
RUST CHINA CONF 2023

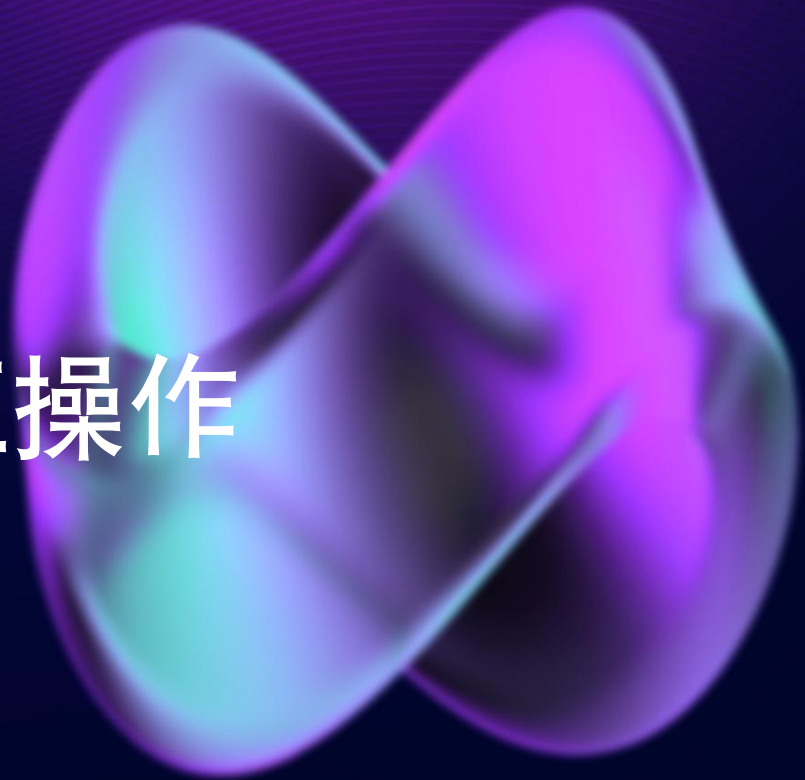
第三届中国Rust开发者大会



6.17-6.18 @Shanghai

SOLANA

在Solana合约链
实现IBC协议跨链互操作



@DaviRain

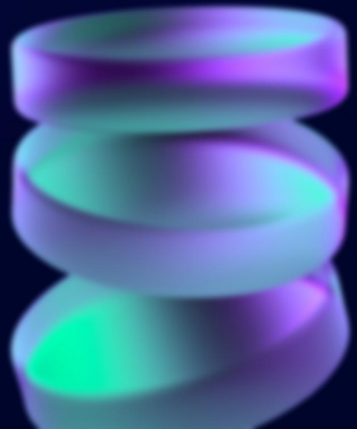
</ 简介 >

简单介绍下IBC协议是什么，及其生态

介绍IBC协议和其在跨链互操作中的作用

IBC协议中的角色和参与者

1. 客户端
2. 连接
3. 通道
4. 包
4. 中继器

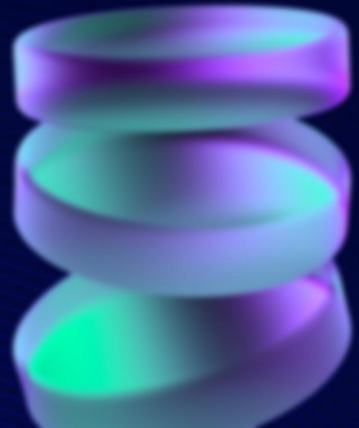


解释为什么选择在Rust合约链中实现IBC协议

- IBC协议的核心已经被协议核心团队用Rust语言实现。

- 对于本身就是使用Rust语言作为智能合约开发的区块链平台来说，支持集成支持IBC协议会很方便。

- 这里优先构想了在Solana链上实现IBC协议，因为Solana平台本身极　低的gas消耗，很适合我现在构思的这套实现方案。（后面会做解释）

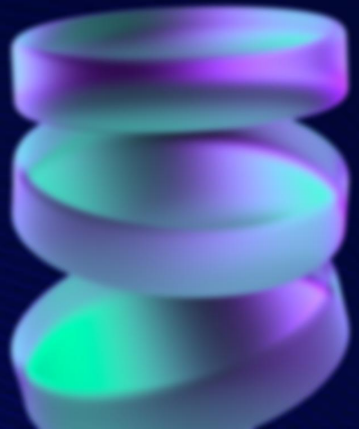


引入Solana作为示例平台

- Solana极低的Gas花销。

- Anchor合约开发框架，大大降低了Rust合约开发者在Solana上开发智能合约的难度。

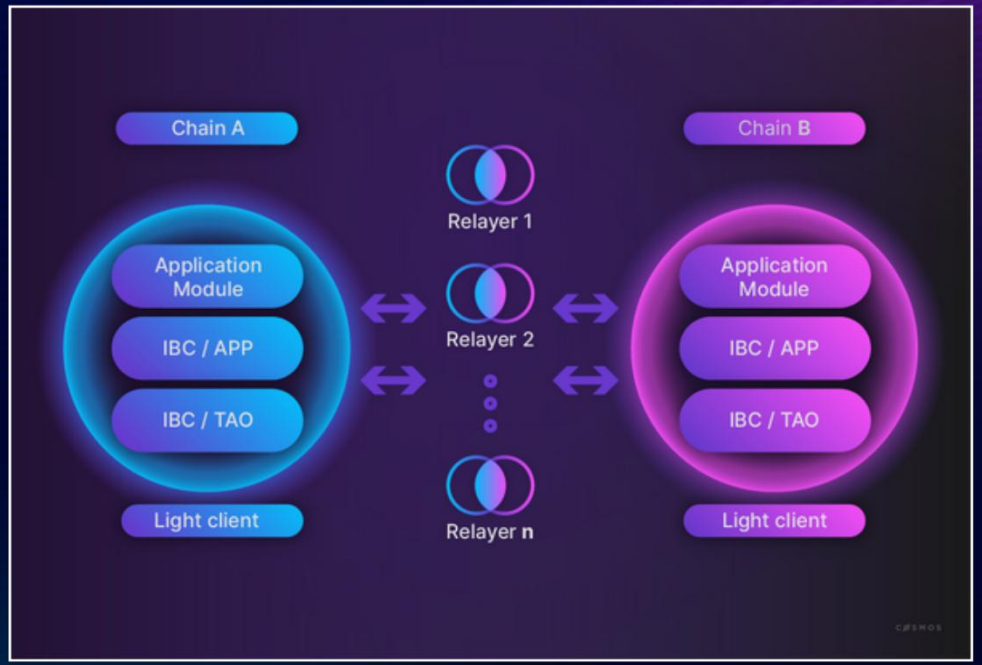
- 以及本人对Solana平台的喜欢，优先考虑了Solana平台，但是这套方案是可以推广到任何的Rust智能合约平台的。



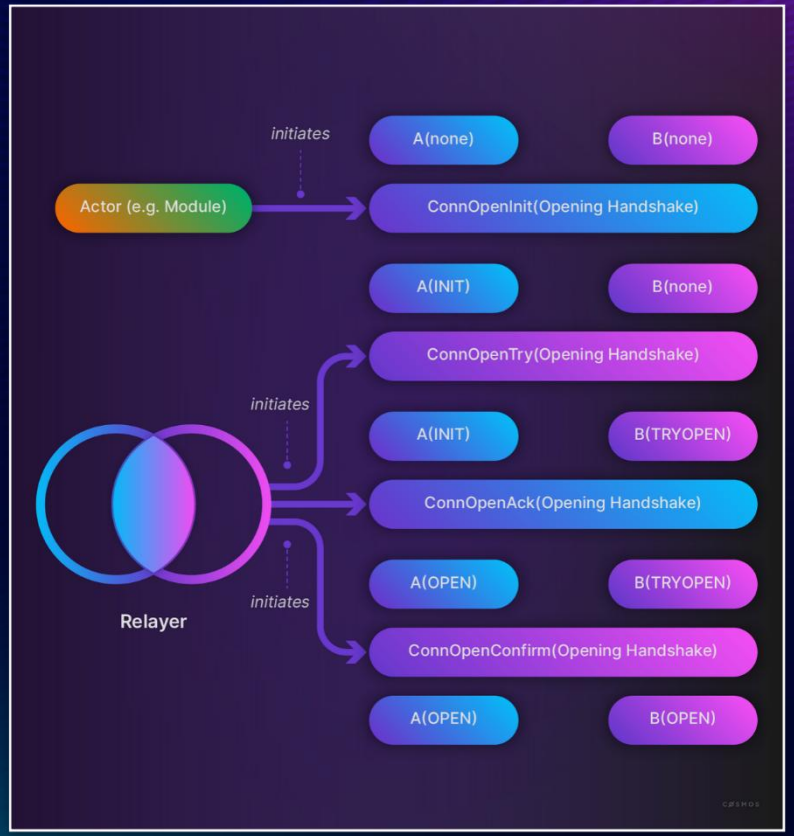
</ IBC协议概述>

大致讲解下IBC协议的原理，
以及参与整个IBC协议活动的不同决策

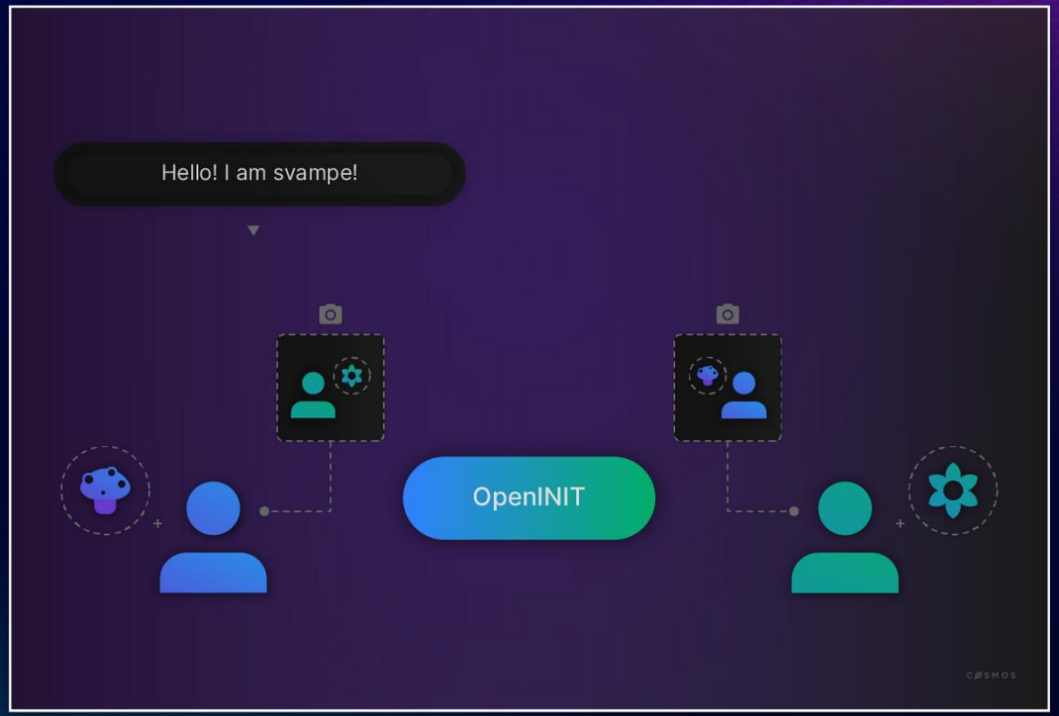
详细介绍IBC协议的基本概念和原理



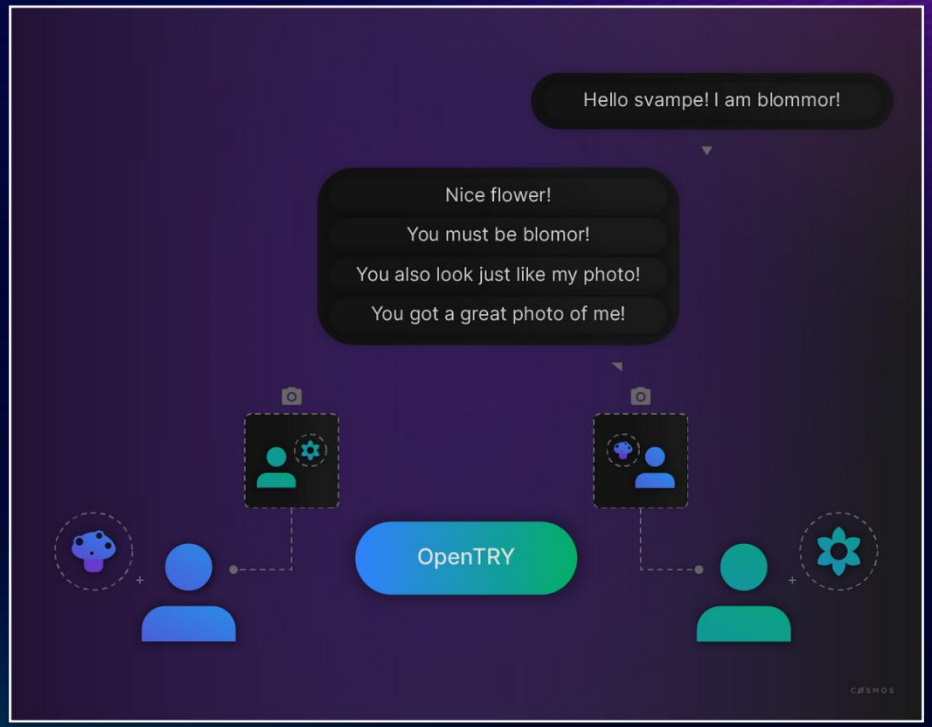
Connection创建原理



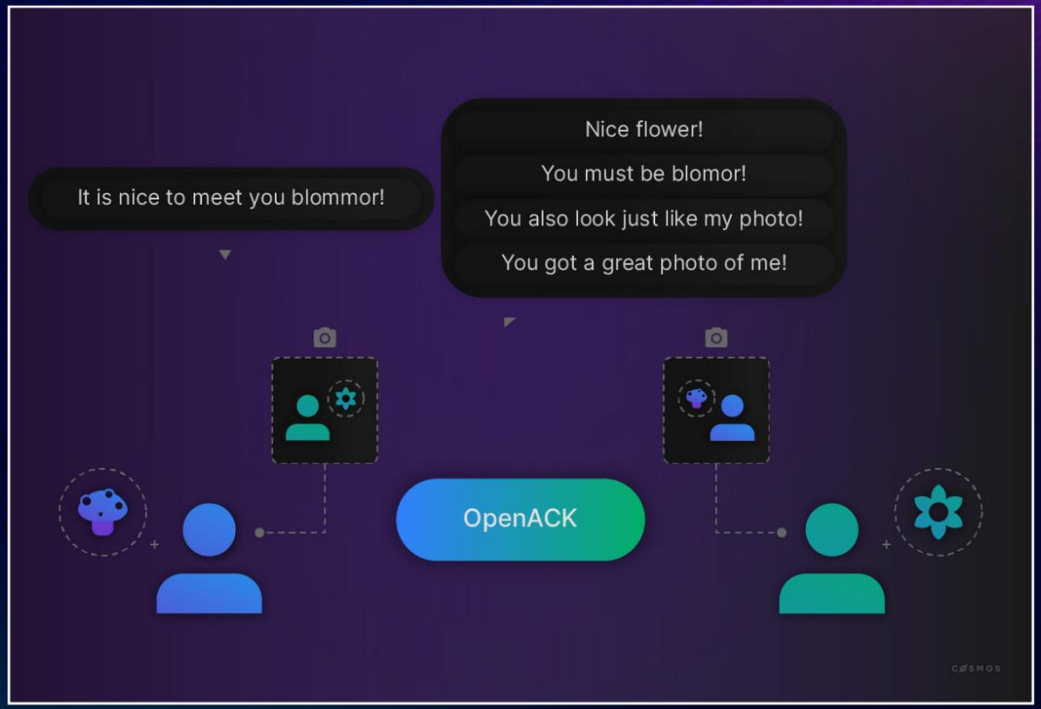
Connection创建OpenInit



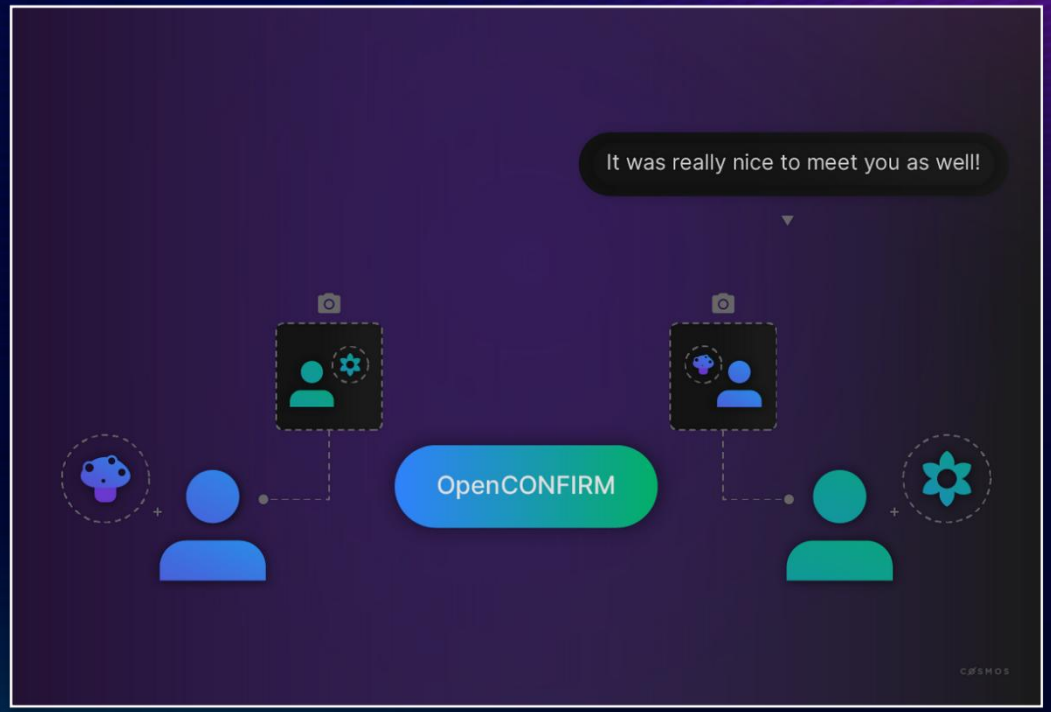
Connection创建OpenTry



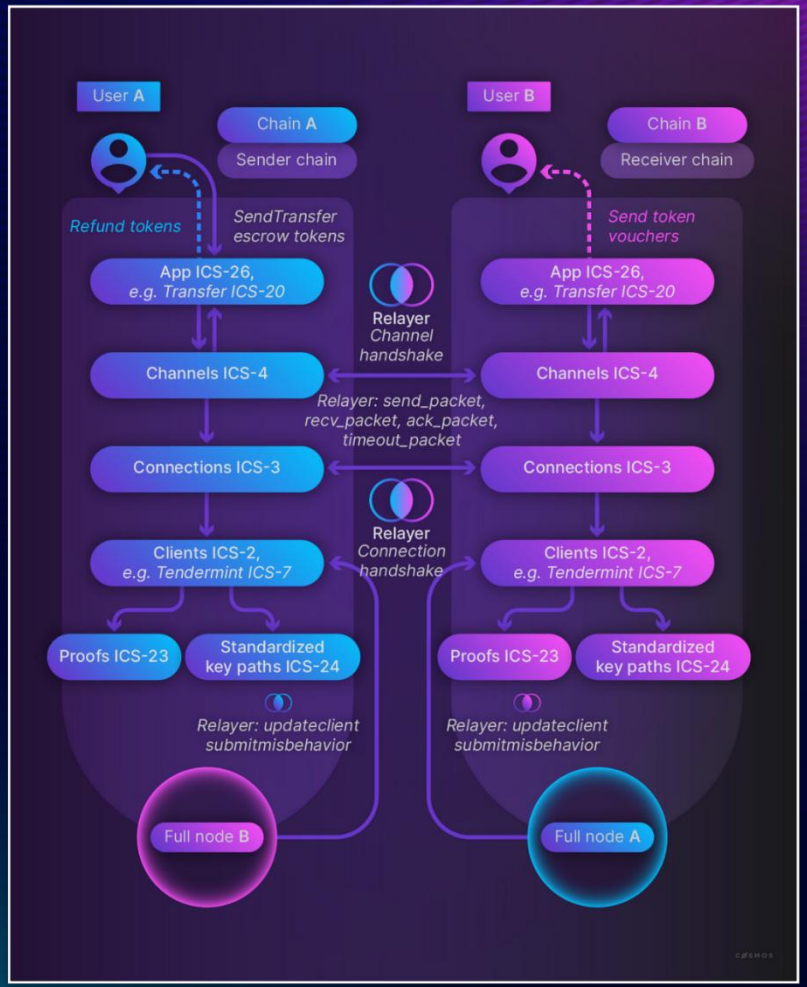
Connection创建OpenAck



Connection创建OpenConfirm



轻客户端核心


User A
Chain A
Sender chain
User B
Chain B
Receiver chain
Refund tokens
SendTransfer
escrow tokens
Send token
vouchers
App ICS-26,
e.g. Transfer ICS-20
App ICS-26,
e.g. Transfer ICS-20
Relayer
Channel
handshake
Channels ICS-4
Channels ICS-4
Relayer: send_packet,
recv_packet, ack_packet,
timeout_packet
Connections ICS-3
Connections ICS-3
Relayer
Connection
handshake
Clients ICS-2,
e.g. Tendermint ICS-7
Clients ICS-2,
e.g. Tendermint ICS-7
Proofs ICS-23
Standardized
key paths ICS-24
Proofs ICS-23
Standardized
key paths ICS-24
Relayer: updateclient
submitmisbehavior
Relayer: updateclient
submitmisbehavior
Full node B
Full node A

</ Rust合约链概述>

使用Rust在Rust合约链实现IBC协议

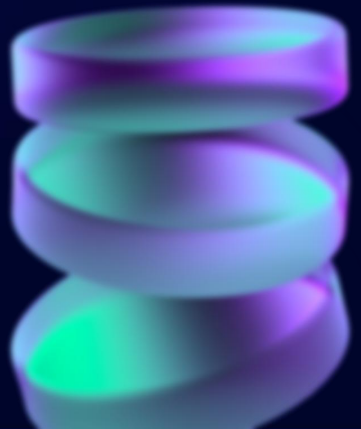
解释为什么选择Rust作为实现IBC协议的语言

1. Rust语言的安全性和性能优势

2. Rust生态系统的丰富性

3.Solana平台的支持：Solana是一个基于Rust开发 的高性能区块链平台，提供了完善的开发工具和文档，可以帮助开发者更加便捷地进行Rust合约链开发。

4. Informal Systems提供的IBC协议Rust语言实现和协议的形式化验证可以有效提高IBC协议的安全性和可靠性，保障跨链交易的安全和正确性。



简要介绍Solana作为使用Rust开发智能合约的平台

```rust
fn process_instruction(
    _program_id: &Pubkey,
    _accounts: &[AccountInfo],
    instruction_data: &[u8],
) -> ProgramResult {
    let (selector, rest) = instruction_data
        .split_first()
        .ok_or(ProgramError::InvalidInstructionData)?;

    match selector {
        0 => msg!(&format!(
            "first instruction called with remain data: {:?}",
            rest,
        )),
        1 => msg!(&format!(
            "second instruction called with remain data: {:?}",
            rest,
        )),
        _ => {
            msg!("invalid called");
            return Err(ProgramError::InvalidInstructionData);
        }
    }

    Ok(())
}
```

```rust
#[program]
pub mod counterapp {
    use super::*;

    pub fn create(ctx: Context<Create>) -> ProgramResult {
        let counter_account = &mut ctx.accounts.counter_account;
        counter_account.count = 0;
        Ok(())
    }

    pub fn increment(ctx: Context<Increment>) -> ProgramResult {
        let counter_account = &mut ctx.accounts.counter_account;
        counter_account.count += 1;
        Ok(())
    }
}

#[derive(Accounts)]
pub struct Create<'info> {
    #[account(init, payer=user, space = 16+16)]
    pub counter_account: Account<'info, CounterAccount>,
    #[account(mut)]
    pub user: Signer<'info>,
    pub system_program: Program<'info, System>,
}

#[account]
pub struct CounterAccount {
    pub count: u64,
}

#[derive(Accounts)]
pub struct Increment<'info> {
    #[account(mut)]
    pub counter_account: Account<'info, CounterAccount>
}
```

</ IBC在Solana上的实现>

在Solana链上实现IBC协议的核心要点

集成ibc-rs仓库实现solana-ibc TAO

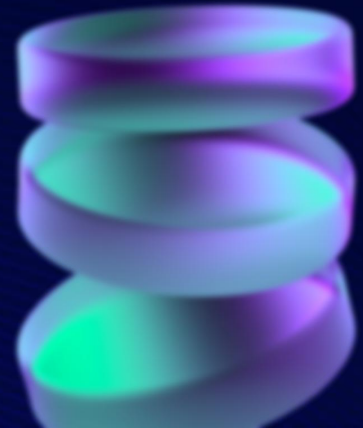
```rust
/// Context to be implemented by the host that provides all "read-only" methods.
///
/// Trait used for the top-level [`validate`](crate::core::validate)
pub trait ValidationContext: Router {
    /// Returns the ClientState for the given identifier `client_id`.
    fn client_state(&self, client_id: &ClientId) -> Result<Box<dyn ClientState>, ContextError>;

    /// Tries to decode the given `client_state` into a concrete light client state.
    fn decode_client_state(&self, client_state: Any) -> Result<Box<dyn ClientState>, ContextError>;

    /// Retrieve the consensus state for the given client ID at the specified
    /// height.
    ///
    /// Returns an error if no such state exists.
    fn consensus_state(
        &self,
        client_cons_state_path: &ClientConsensusStatePath,
    ) -> Result<Box<dyn ConsensusState>, ContextError>;
```

集成ibc-rs仓库实现Solana-ibc TAO

```rust
/// Context to be implemented by the host that provides all "write-only" methods.
///
/// Trait used for the top-level [`execute`](crate::core::execute) and [`dispatch`](crate::core::dispatch)
pub trait ExecutionContext: ValidationContext {
    /// Called upon successful client creation and update
    fn store_client_state(
        &mut self,
        client_state_path: ClientStatePath,
        client_state: Box<dyn ClientState>,
    ) -> Result<(), ContextError>;

    /// Called upon successful client creation and update
    fn store_consensus_state(
        &mut self,
        consensus_state_path: ClientConsensusStatePath,
        consensus_state: Box<dyn ConsensusState>,
    ) -> Result<(), ContextError>;

    /// Called upon client creation.
    /// Increases the counter which keeps track of how many clients have been created.
    /// Should never fail.
    fn increase_client_counter(&mut self);
```

在Relayer侧（中继器）需要实现的对应链的创建查询接口

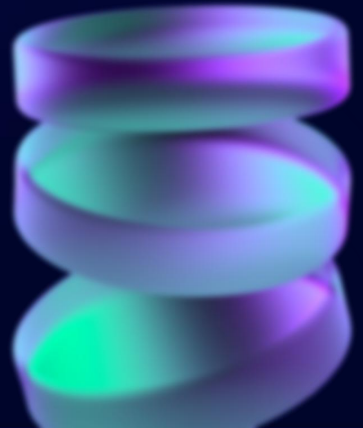
```rust
/// Defines a blockchain as understood by the relayer
pub trait ChainEndpoint: Sized {
    /// Type of light blocks for this chain
    type LightBlock: Send + Sync;

    /// Type of headers for this chain
    type Header: Header + Into<AnyHeader>;

    /// Type of consensus state for this chain
    type ConsensusState: ConsensusState + Into<AnyConsensusState>;

    /// Type of the client state for this chain
    type ClientState: ClientState + Into<AnyClientState>;

    /// The type of time for this chain
    type Time;

    /// Type of the key pair used for signatures of messages on chain
    type SigningKeyPair: SigningKeyPairSized + Into<AnySigningKeyPair>;

    /// Returns the chain's identifier
    fn id(&self) -> &ChainId {
        &self.config().id
    }
```



为什么能将所有Rust合约链看作ICS06 Solomachine Client

这里就是整个方案的核心，
为什么我能将所有的rust合约链能看作是ICS06 solomachine cleint

- 安全性保障的方案。
    - ICS06 Solomachine Client 本身支持单个签名和多个签名
    - 单个签名肯定是不安全的，需要引入多个签名的方案，
    - 但是多个签名也是会出现内部作恶的，虽然是通过Dao选举出来的，通过多个签名成员不会作恶保证。
    - 改进方案，通过引入签名人池，有资格签名的人进入签名人池，之后通过合约随机选取部分签名人，只要这一部分签名人的1/3即可。这种方法保证了随机公平性（这里的公平性需要更好的随机算法实现）。创新的核心就是通过随机选择签名人。进而这个方案可以快速的推广到更多不同共识算法的Rust合约链（Substrate chain，Near，Solana，Oasis，Nerovs 等等）实现。



</ 总结和展望 >

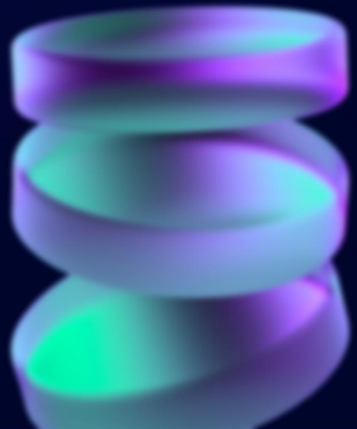
总结，以及对未来IBC生态的展望

提供代码示例或参考链接以帮助读者更好地理解实现细节

Pallet-ibc 的参考实现：
https://github.com/DaviRain-Su/pallet-ibc

Solana-ibc的参考实现（未开发完整）：
https://github.com/DaviRain-Su/solana-ibc

Hermes的开发实现：
https://github.com/DaviRain-Su/hermes



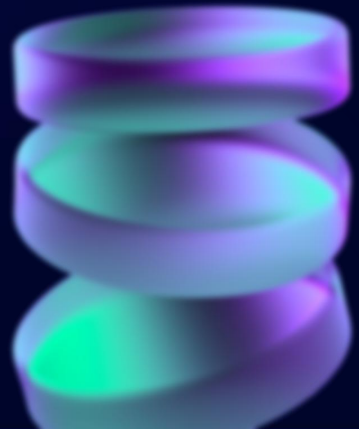
展望IBC协议在跨链互操作中的未来发展

1. 基于IBC协议的全链去中心化交易所（已经有了），衍生资产市场等。

2. 基于IBC协议的全链应用（类似WeChat 或许会出现）

3. 更加高效的支持跨链互操作，随着协议的不断发展与进化。

4. 更多的区块链平台的支持。（例如这里的这套方案，可以将任何链的共识看作 是ICS06类似的共识处理，支持更多的Rust合约链集成IBC协议）

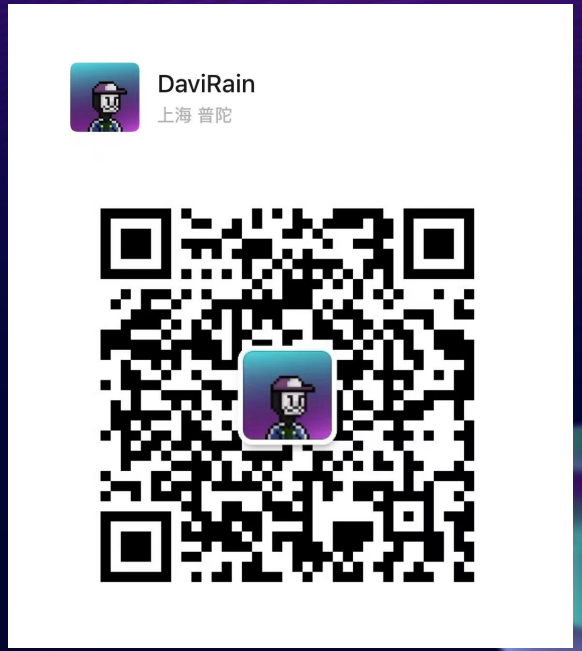


</ 相关资料 >

有关IBC协议的相关资料

有关与IBC协议相关的资料

- IBC协议的Sepc文档
- IBC协议的Rust语言实现
- IBC协议的Go语言实现
- IBC协议使用Solana合约实现(working)
- IBC协议的Substrate-ibc实现
- IBC协议的ICS06 Rust语言实现
- IBC协议的Relayer(Hermes)
- IBC协议的官网



Thank you!