

# Presentation Title

## Optional Subtitle

K. Cong

Faculty of Electrical Engineering, Mathematics and Computer Science  
Delft University of Technology

2017

# Outline

## Background

- TrustChain

## My Thesis

- TrustChain with Checkpoints

- Protocol Overview

- Promoter Registration

- Consensus

- Validation

- Implementation

# Outline

## Background

TrustChain

## My Thesis

TrustChain with Checkpoints

Protocol Overview

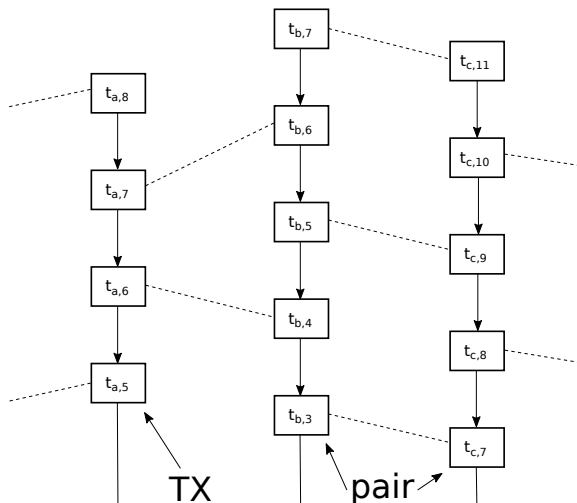
Promoter Registration

Consensus

Validation

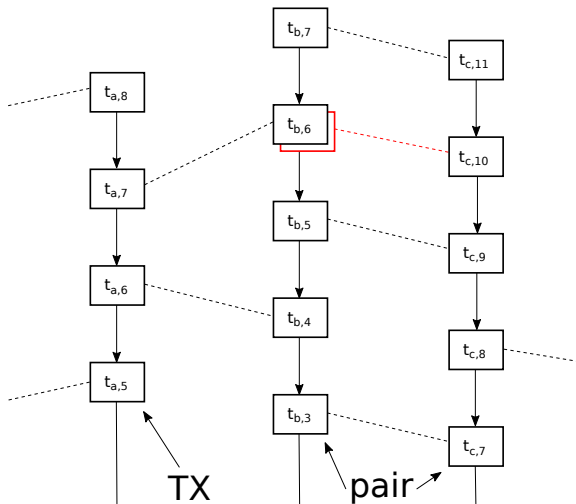
Implementation

# TrustChain



**Figure:** Every node has a chain. *TX block* is a six-tuple:  
 $t_{i,j} = (h(b_{i,j-1}), h_s, h_r, s_s, s_r, m)$ , one transaction results in two TX blocks—a *pair*.

# TrustChain



**Figure:** Fork is two correctly signed TX blocks that belong to the same sender and has the same hash pointer but involve different receivers. Only one TX block may be in consensus.

# TrustChain

- ▶ Everyone has their own chain
- ▶ Transactions are on arbitrary data  $m$
- ▶ Transactions are irrefutable due to hash pointers
- ▶ No consensus (my thesis)

# Outline

## Background

TrustChain

## My Thesis

TrustChain with Checkpoints

Protocol Overview

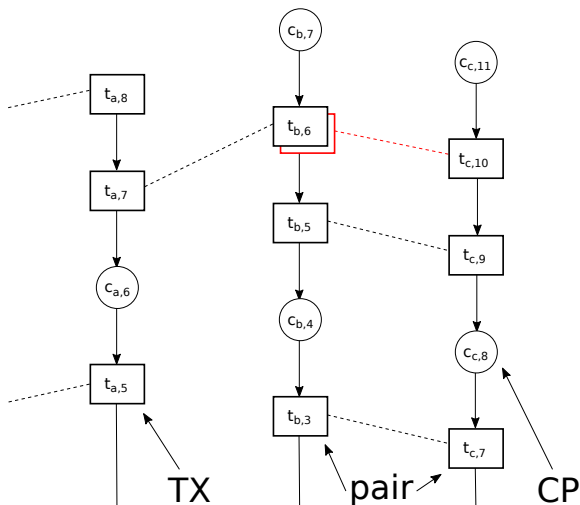
Promoter Registration

Consensus

Validation

Implementation

# TrustChain with Checkpoints



**Figure:** CP block is a five-tuple:  $c_{i,j} = (\mathbf{h}(b_{i,j-1}), \mathbf{h}(\mathcal{C}_r), r, p, s)$ ,  $\mathcal{C}_r$  is the consensus result at round  $r$ ,  $p$  = promoter indicator,  $s$  = signature.



# Outline

## Background

TrustChain

## My Thesis

TrustChain with Checkpoints

**Protocol Overview**

Promoter Registration

Consensus

Validation

Implementation

# Protocol Overview

1.  $n$  lucky nodes are selected at random to act as promoters.
2. Promoters run a BFT (Byzantine Fault Tolerant) consensus algorithm to agree on a set of CP blocks.
3. Disseminate the consensus result (the CP blocks).
4. Repeat for next round.
5. Any interested node can validate that their transaction.

# Outline

## Background

TrustChain

## My Thesis

TrustChain with Checkpoints

Protocol Overview

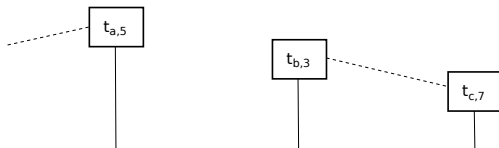
**Promoter Registration**

Consensus

Validation

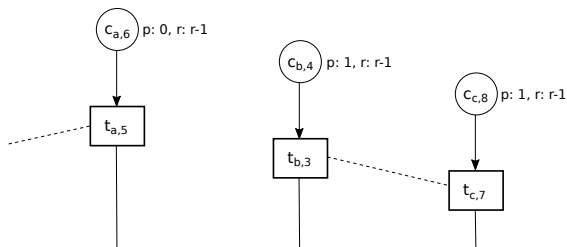
Implementation

# Promoter Registration



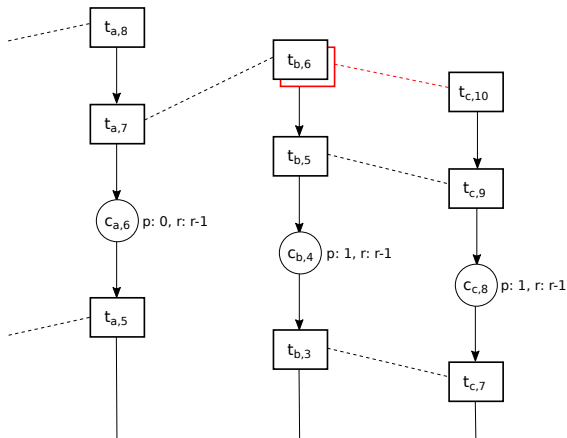
**Figure:** We start in the state where  $\mathcal{C}_{r-1}$  has just been agreed but not yet propagated.

# Promoter Registration



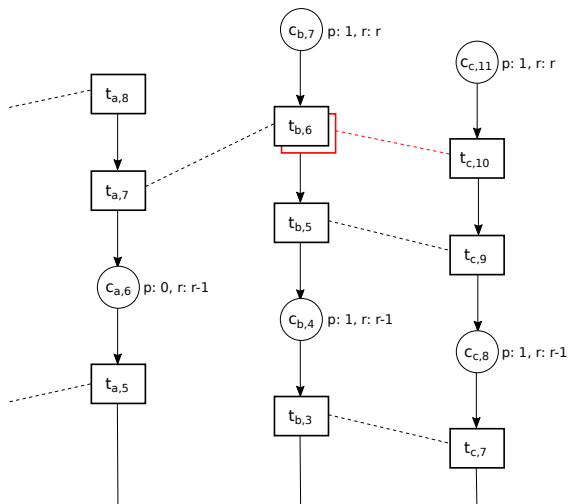
**Figure:** Nodes receive consensus result and set promoters indicator  $p$ , then send the new CP blocks to promoters of round  $r$ .

# Promoter Registration



**Figure:** Transactions carry on as usual in round  $r$ . Note that our CP blocks (round  $r - 1$ ) has not reached consensus yet.

# Promoter Registration



**Figure:** CP blocks at round  $r - 1$  should be in  $\mathcal{C}_r$ . If we're lucky:  $h(k_i || c_{i,j}) < T$ , then we're responsible for consensus of round  $r + 1$ .

# Outline

## Background

TrustChain

## My Thesis

TrustChain with Checkpoints

Protocol Overview

Promoter Registration

**Consensus**

Validation

Implementation



# Consensus

1. Nodes send CP blocks to the promoters.
2. The promoters' identities are encoded in the consensus result.
3. Promoters run the same asynchronous BFT consensus algorithm to agree on a set of CP blocks— $\mathcal{C}_r$ .
4.  $n \geq 3t + 1$  is the optimal for BFT consensus.
5.  $\mathcal{C}_r$  and the signatures are disseminated.
6. Nodes create new CP blocks when they receive  $t + 1$  good signatures and  $\mathcal{C}_r$ .

## Theorem

*Assuming the promoters satisfy  $n \geq 3t + 1$ . The promoter registration and the consensus protocol satisfies agreement, total order and liveness.*

# Outline

## Background

TrustChain

## My Thesis

TrustChain with Checkpoints

Protocol Overview

Promoter Registration

Consensus

**Validation**

Implementation

# Validation

Assume node  $u$  is aware of all the past consensus results  $\mathcal{C}_r$ . Suppose  $u$  wish to validate  $t_{i,j}$ . It performs the following.

1. Determine the pair  $t_{i',j'}$ .
2. Find the agreed enclosure for  $t_{i,j}$  and  $t_{i',j'}$  from  $\mathcal{C}_r$ , otherwise return “unknown”.
3. Query  $i$  and  $i'$  for the agreed pieces and ensure hash pointers are correct. Otherwise return “unknown”.
4. Check that  $t_{i,j}$  and  $t_{i',j'}$  are in the agreed pieces and are created correctly using `newtx`. Otherwise return “invalid”.
5. Check the checkpoints  $c_{i,k}$  and  $c_{i',k'}$  that immediately follow  $t_{i,j}$  and  $t_{i',j'}$  are in the agreed pieces and are created in the same round, i.e.  $\text{round}(c_{i,k}) = \text{round}(c_{i',k'})$ . Otherwise return “invalid”.
6. Return “valid”.

# Validation

In essence, given a TX, ask the receiver to proof that it has a set of transactions that produces some CP block, the CP block should be in the consensus result and the pair of the TX should be in that set.

## Theorem

*If at least one party of every transaction is honest, then forking and other forms of tampering is guaranteed to be detected if the malicious party is alive.*

## Question

Given a starting digest and an ending digest, is it possible for the prover to proof to the verifier that it has a set of blocks linked by hash pointers that “fits” inside those digests in *zero knowledge*.

# Outline

## Background

TrustChain

## My Thesis

TrustChain with Checkpoints

Protocol Overview

Promoter Registration

Consensus

Validation

**Implementation**

# Implementation

- ▶ Currently on going—using Python and Twisted
- ▶ Completed BFT consensus algorithm
- ▶ Completed local TrustChain with Checkpoints
- ▶ Next step is networked TrustChain and validation protocol