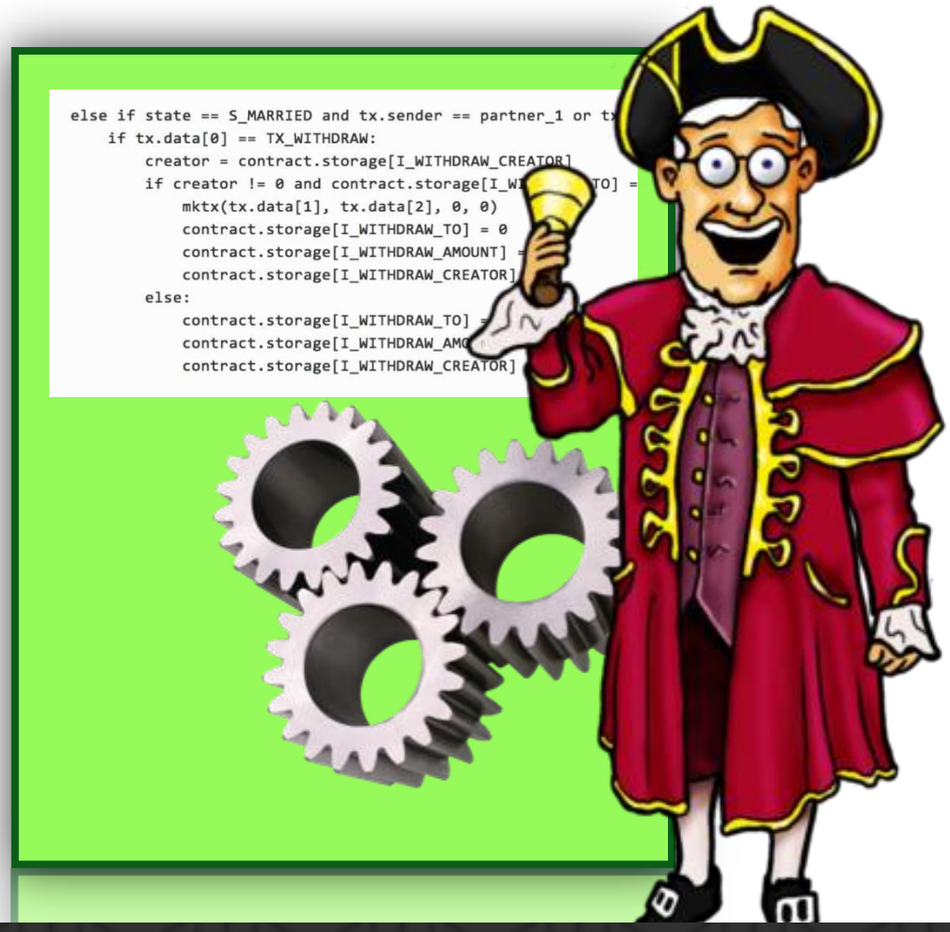
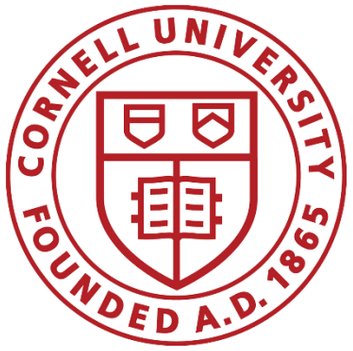
Town Crier:
An Authenticated Data Feed for Smart Contracts

Author: Fan Zhang

Pre: Dinghao Liu





IC3 The Initiative For CryptoCurrencies & Contracts

What's a (decentralized) smart contract?

- Executable object on blockchain
- Programed in Turing-complete languages
- Code can define arbitrarily rich transaction logic
- Decentralized → autonomous

Execution enforced by network

Abstraction: Smart contract is virtual *trusted third party with *public* state.*

Running example:
Self-enforcing **flight delay insurance**

Flight Insurance

Gimme a policy for **Flight #1234** on **17 May** for a **policy price $1**.

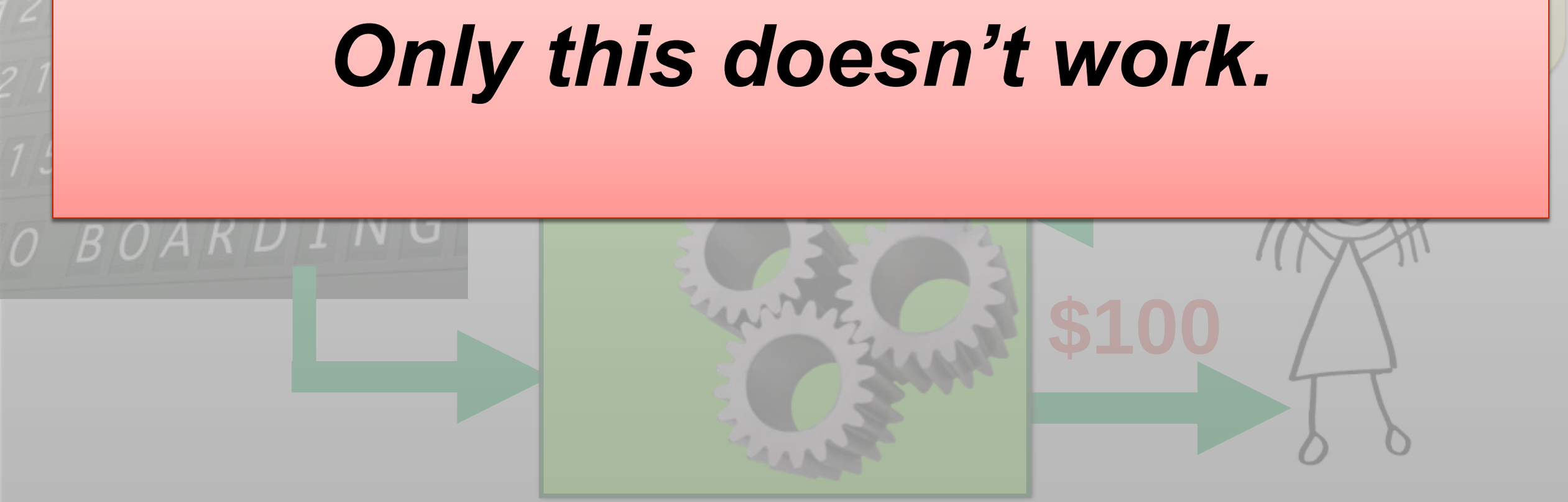
$1

$100

Virtual Trusted Third Party

Gimme a policy for

Only this doesn't work.



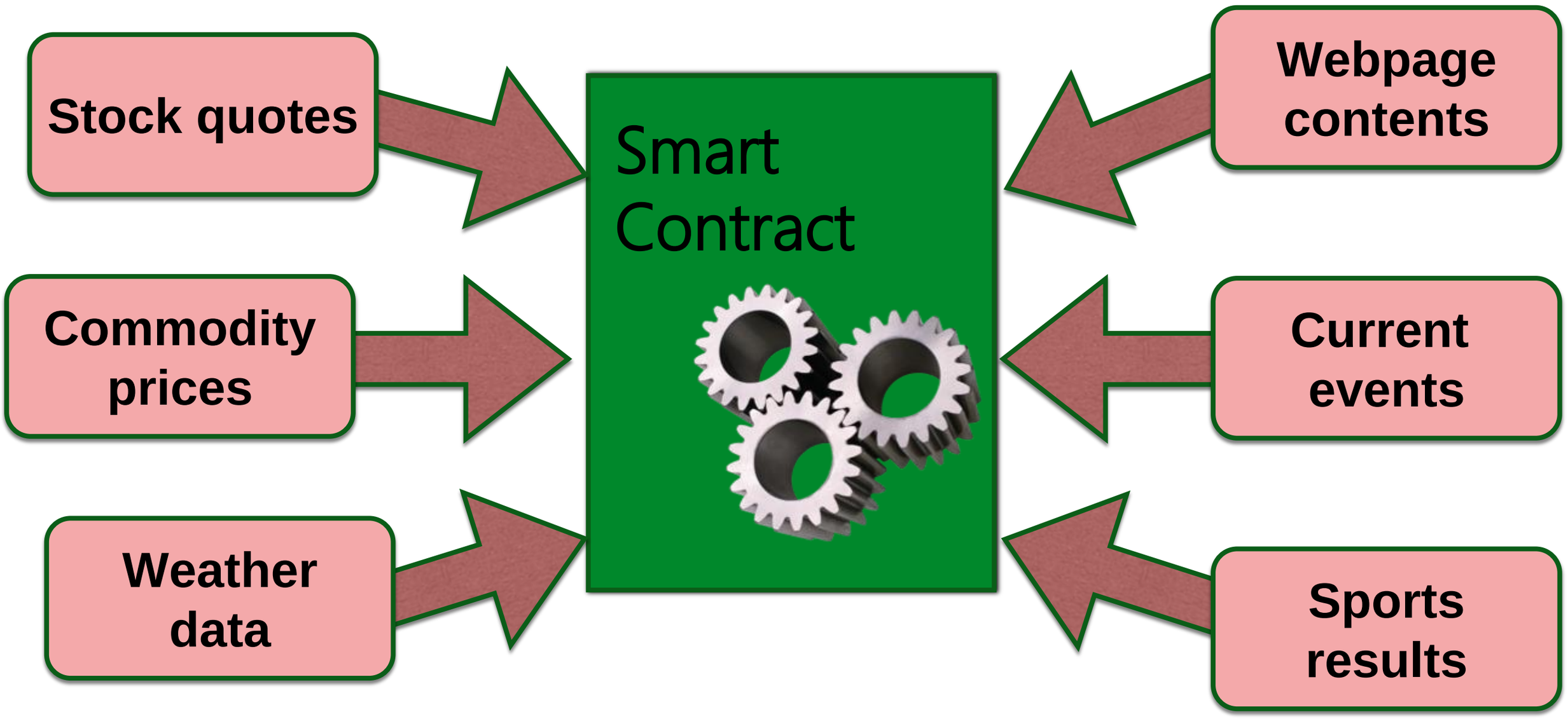
Smart contracts are data-hungry, but…

Flight Insurance

Gimme a policy for **Flight #1234** on **17 May** for a **policy price $1**.

$100

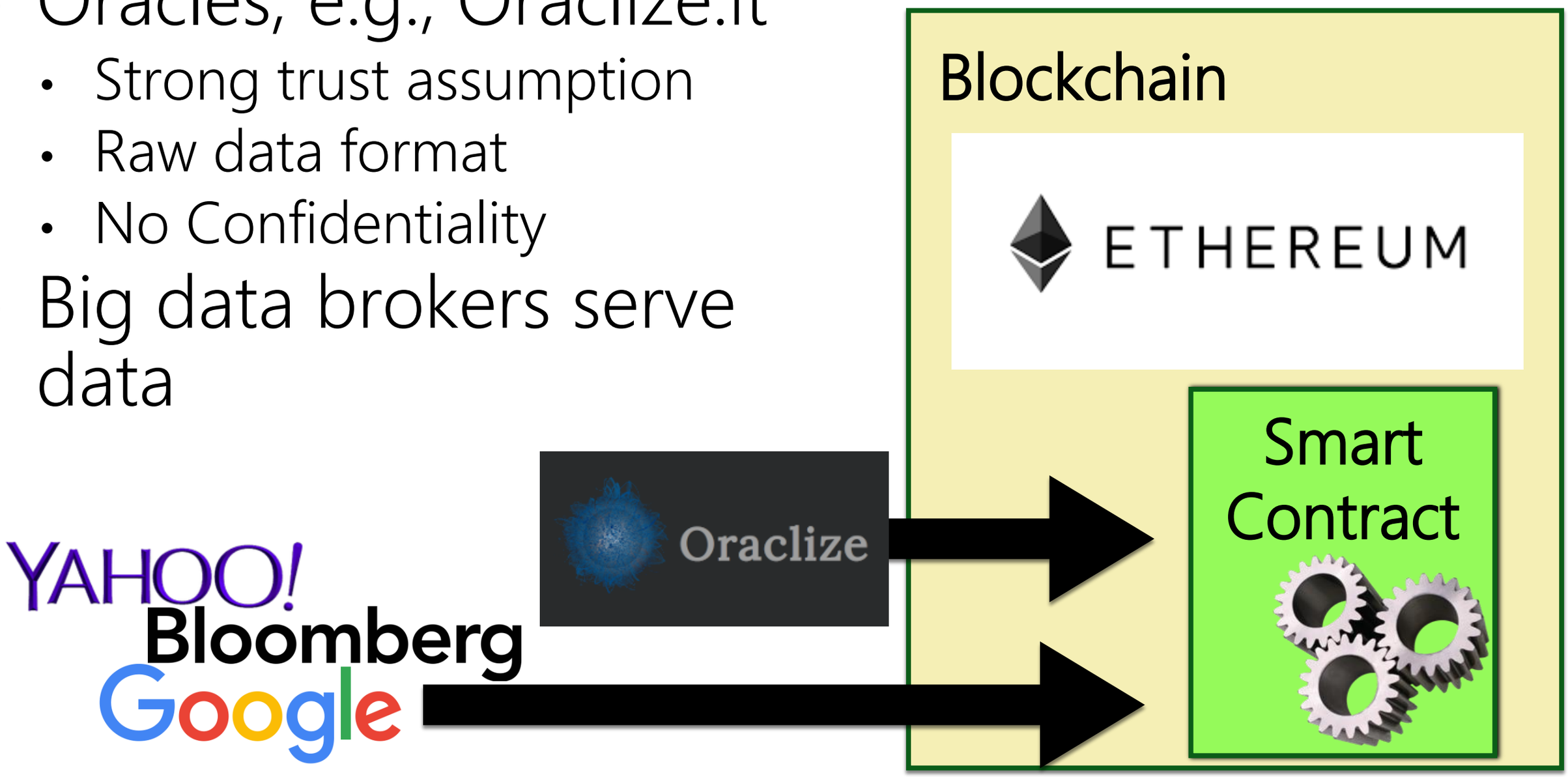
Interesting smart contracts need ***trustworthy data***



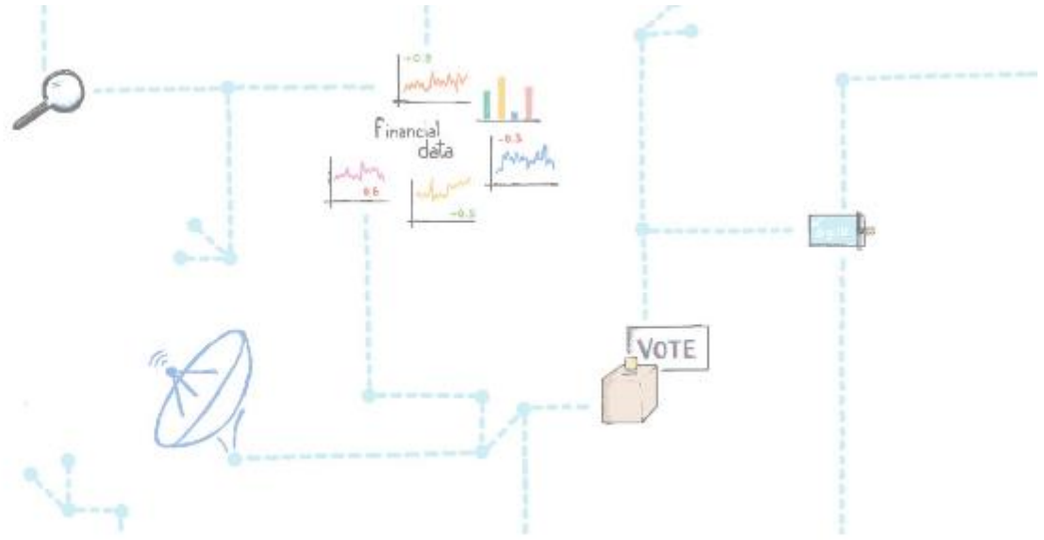
Today there are no good sources.

Proposed data-delivery approaches

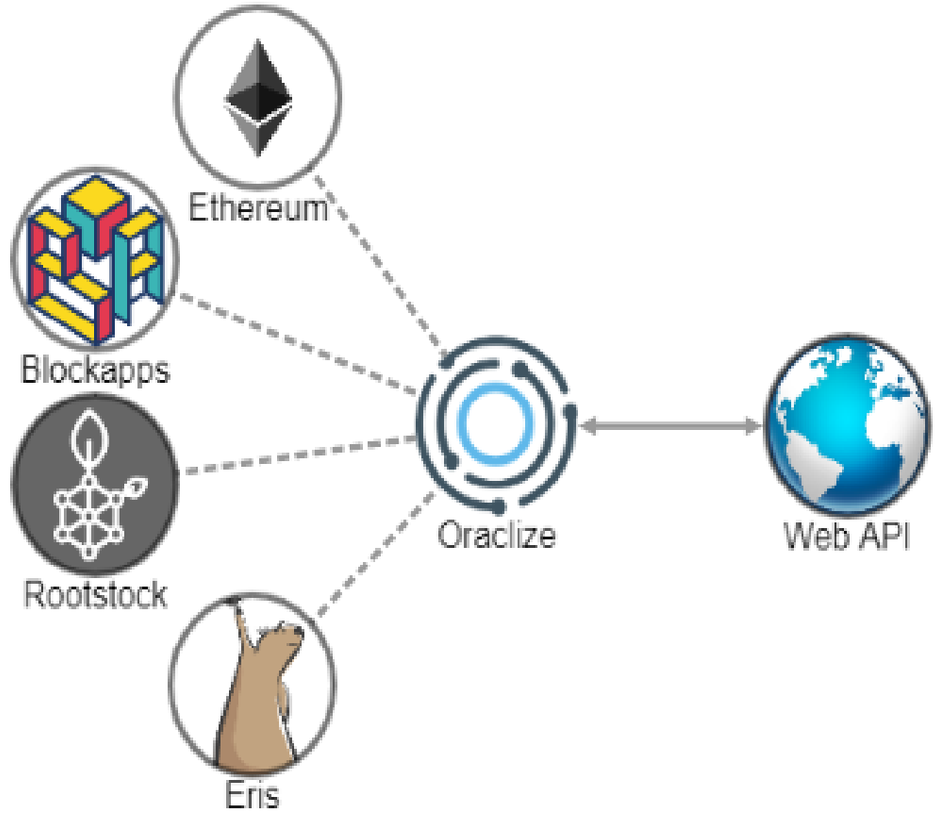
- Prediction markets (e.g., Gnosis)
- Oracles, e.g., Oraclize.it
  - Strong trust assumption
  - Raw data format
  - No Confidentiality
- Big data brokers serve data



Oraclize



- HOW IT WORKS
- *data carrier for decentralized apps*
- Smart contracts live like in a walled garden, they cannot fetch external data on their own. Oraclize is here to help. We act as a data carrier,
  a reliable connection between Web APIs and your Dapp.
  There is no need to open additional trustlines as
  our good behaviour is enforced by cryptographic proofs.



Town Crier

Fan Zhang, Ethan Cecchetti, Kyle Croman,
Elaine Shi, and Ari Juels. Town Crier: An Authenticated Data Feed for Smart Contracts. ACM CCS. 2016.

Town Crier (TC): Basic Idea

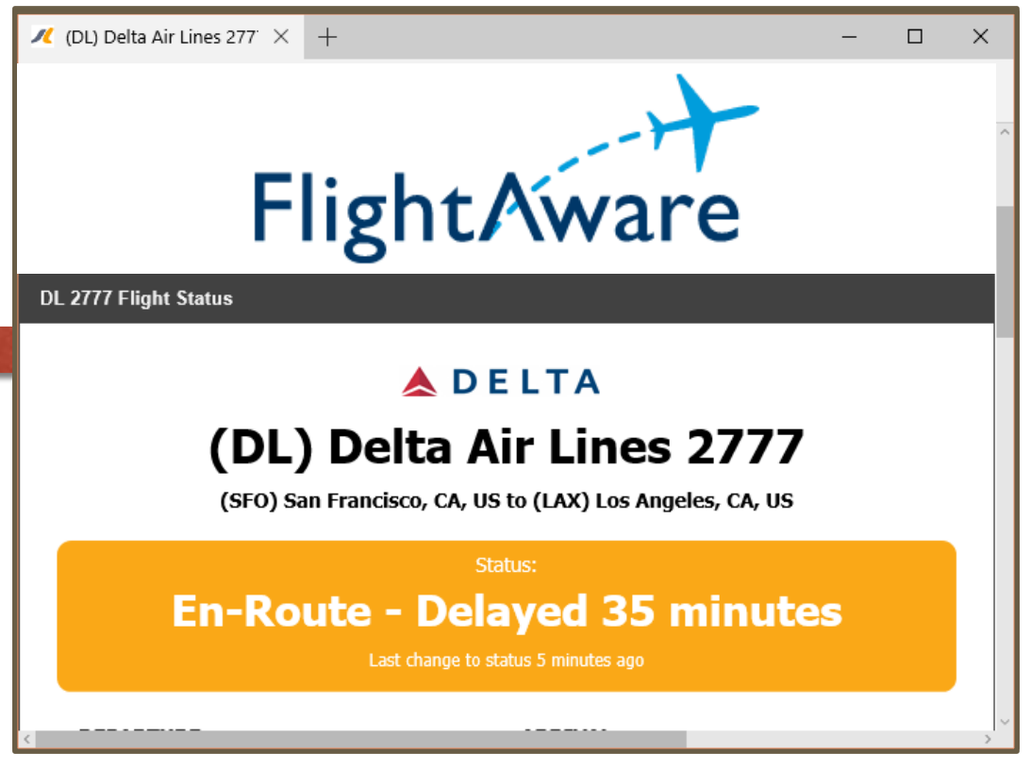
Blockchain

Flight Insurance

Is DL 2777 delayed?

Town Crier

Trusted Website
XYZ.com

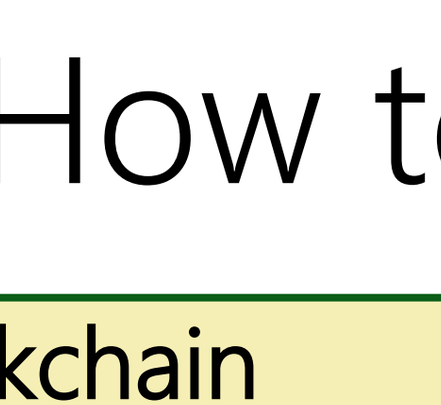


Authenticity property: Data delivered by TC is exactly as served on source site XYZ.com

How to ensure authenticity?

Blockchain

Flight Insurance



intel SGX

Trusted Website
XYZ.com

Intel SGX: Isolation

Integrity

Other processes and even OS cannot tamper with control flow of X

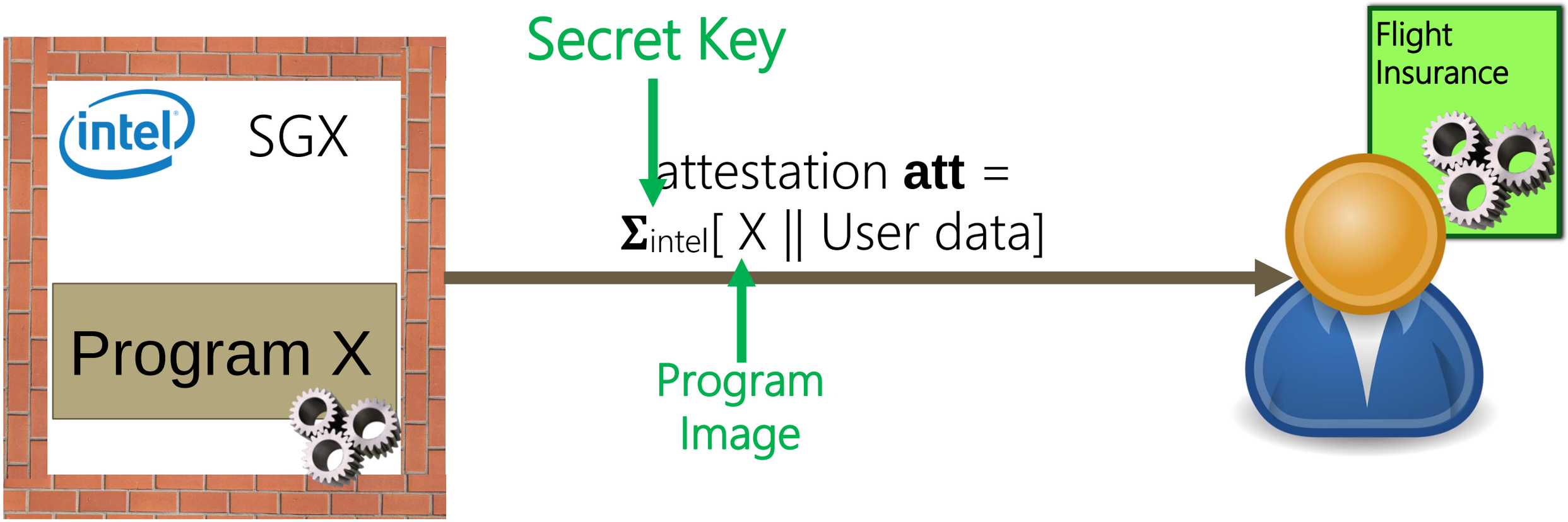
Enclave

Confidentiality

Other processes and even OS can learn nothing* about the state of X

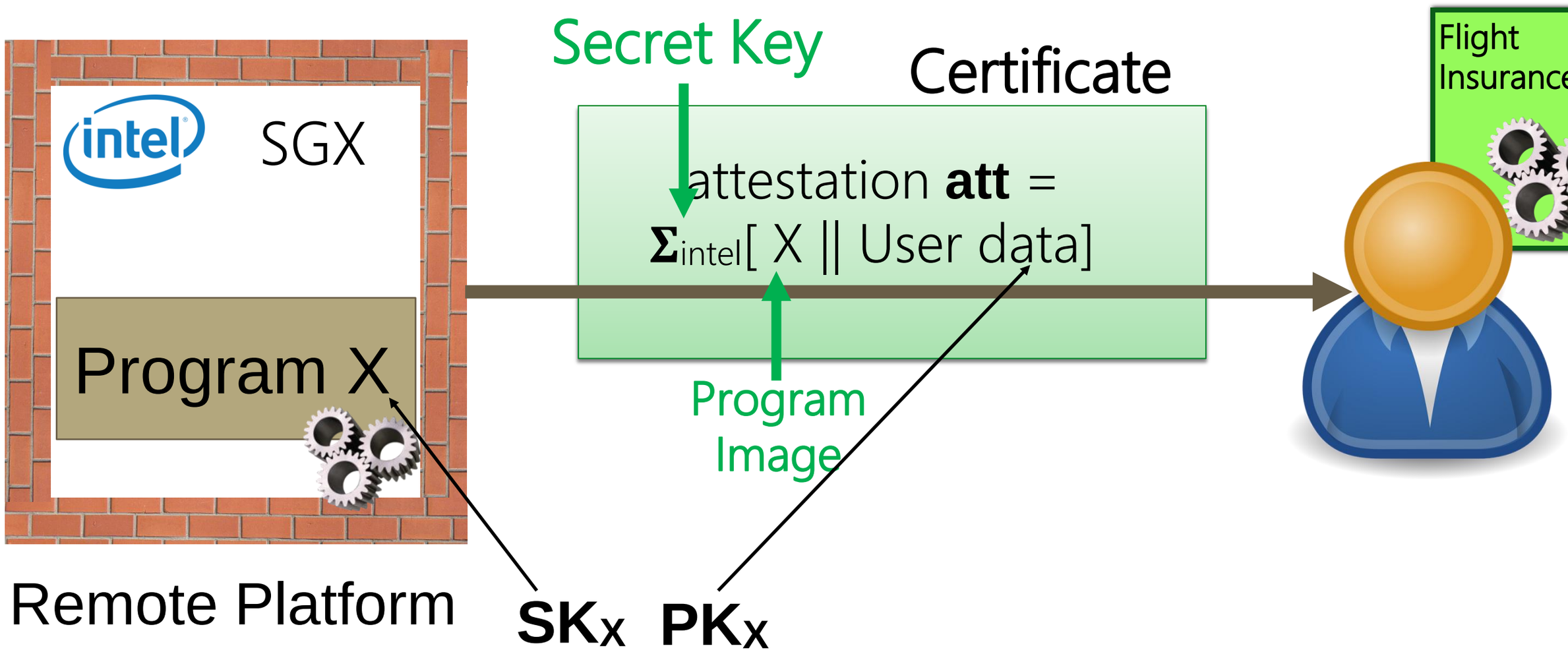
** Side-channels like page faults excepted*

Intel SGX: Remote attestation



**Signature $\boldsymbol{\Sigma}$ (EPID) can be anonymous (group) or pseudonymous*

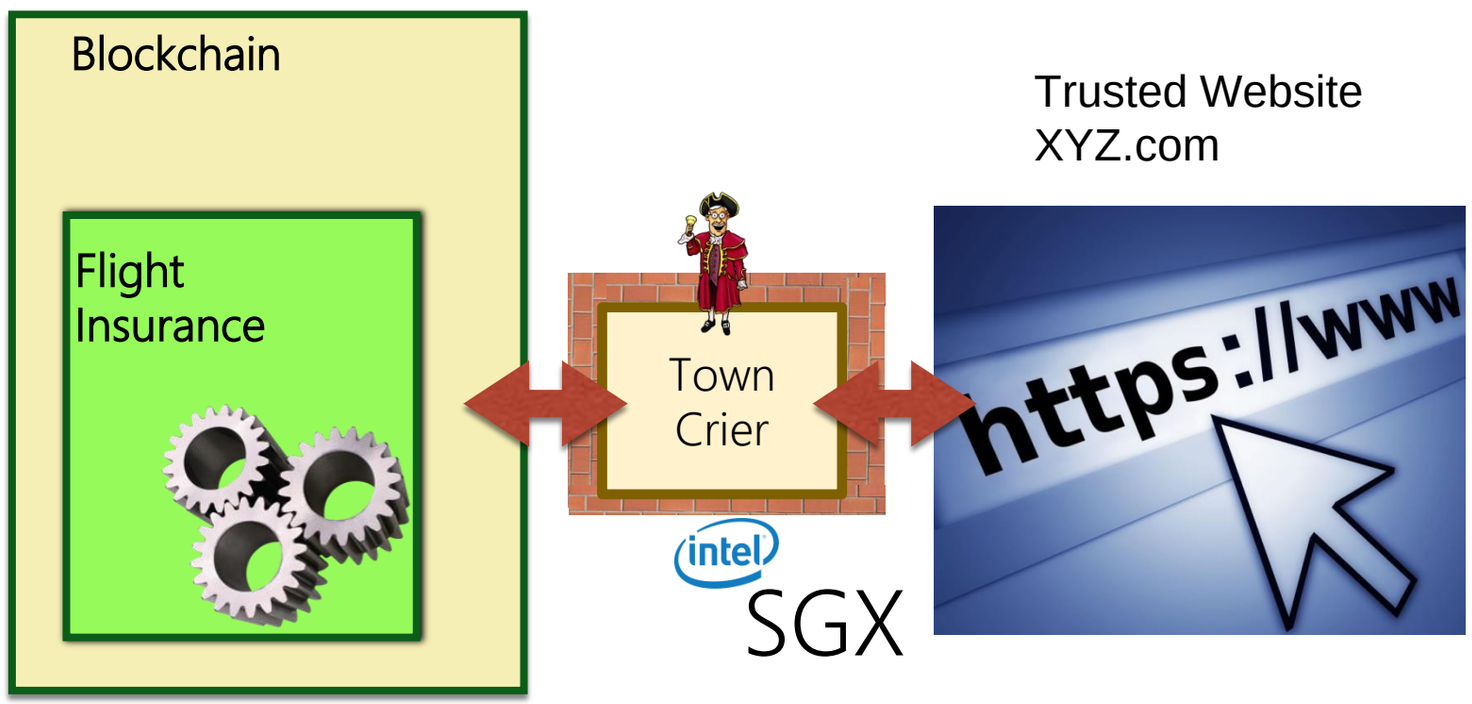
Intel SGX: Secure Channel



**Signature Σ (EPID) can be anonymous (group) or pseudonymous*

TC goal / adversarial model

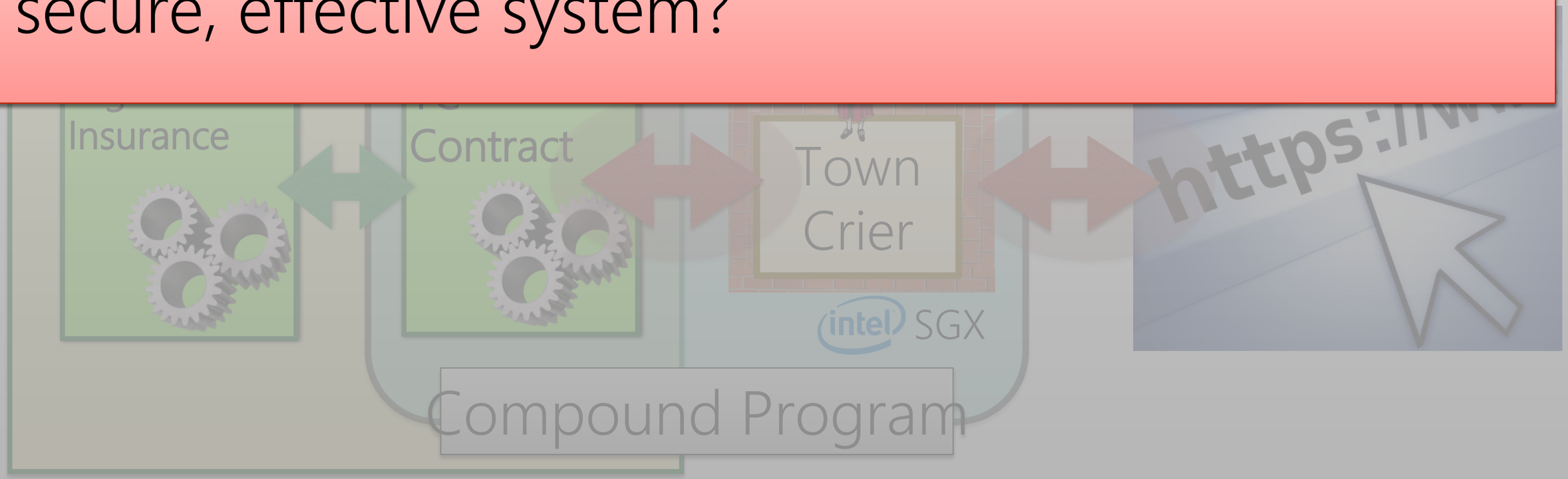
- Relying contract sends query Q = (XYZ, params, $\vec{T}$) to TC
- Goal: TC returns correct answer A to query Q
- Adversary controls the OS of TC server and the network
  - Simpler view: **adversary controls network outside enclave**



Tripartite trust model

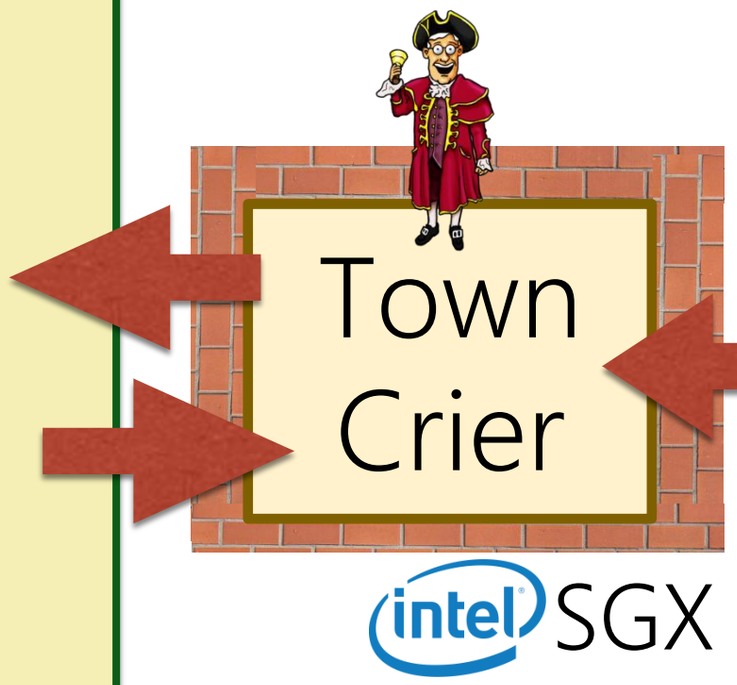
- **Smart contract**
  - Trustworthy execution
  - No confidentiality
- **TC enclave**
  - Trustworthy execution
- **Trusted website**
  - Source authentication

Key Challenge: How does one stitch together disparate trust domains with different properties into a secure, effective system?

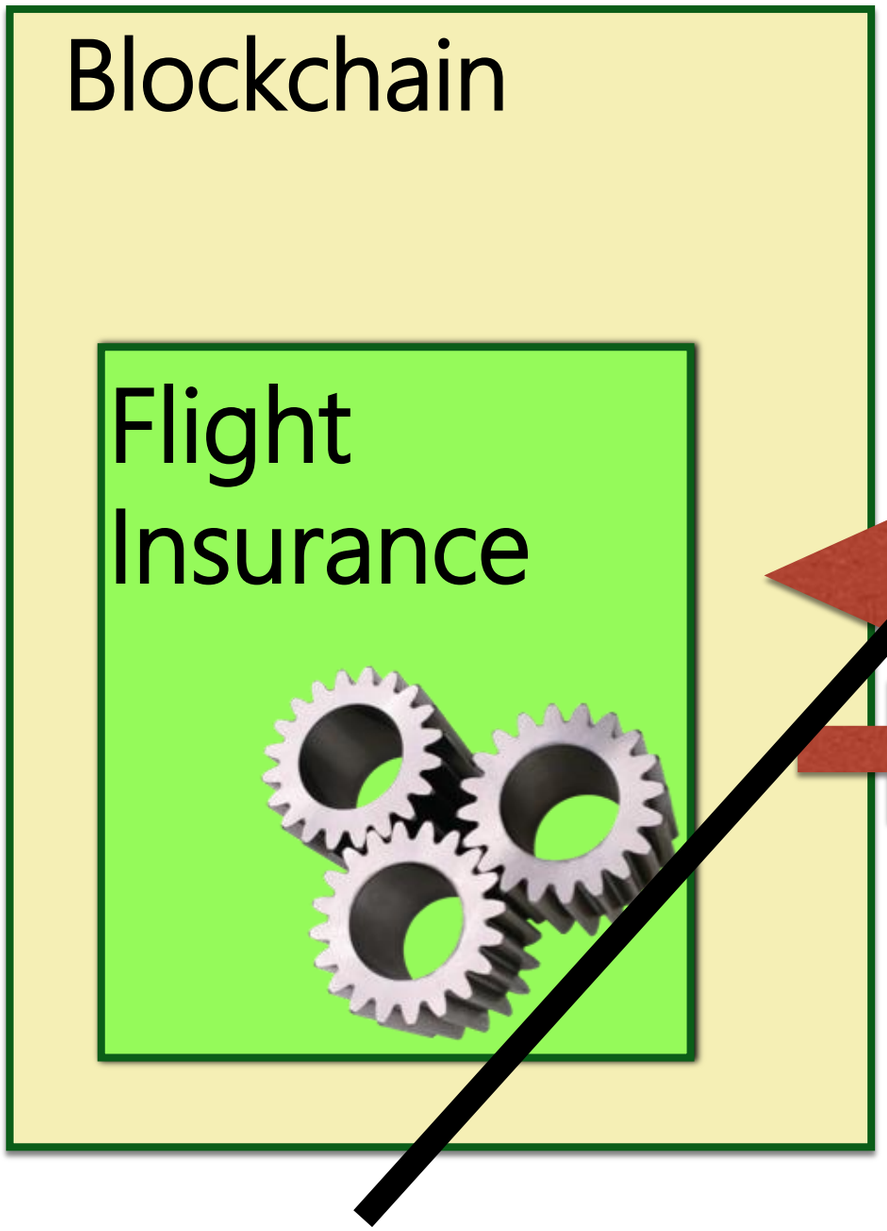


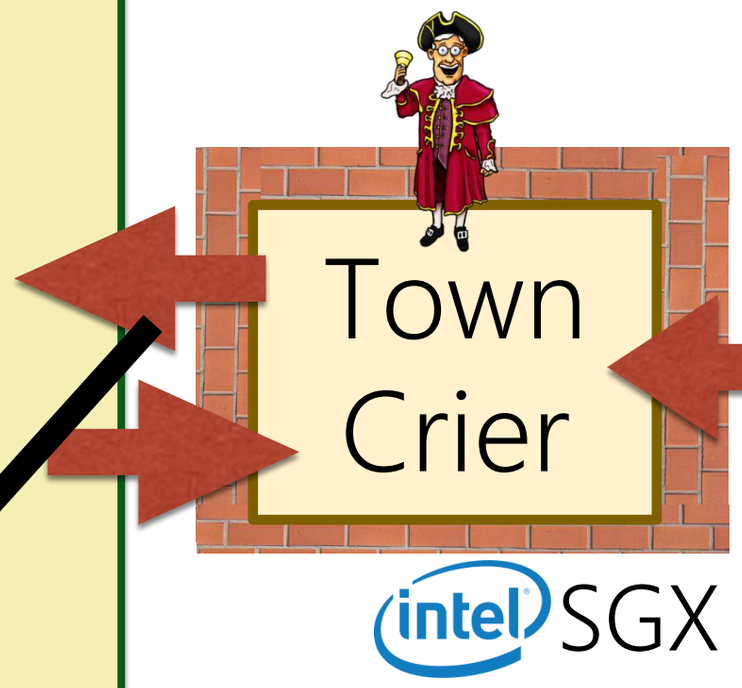
Blockchain

Flight Insurance



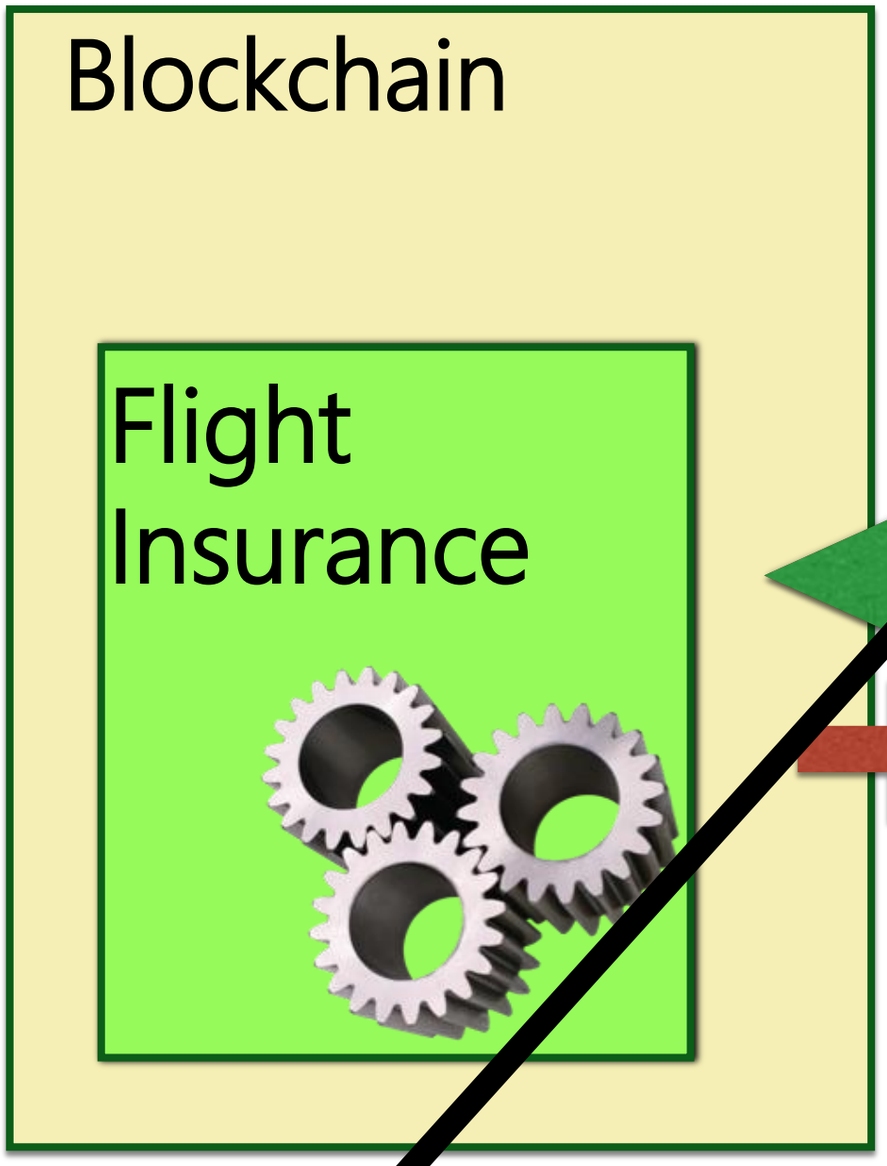
Trusted Website
XYZ.com

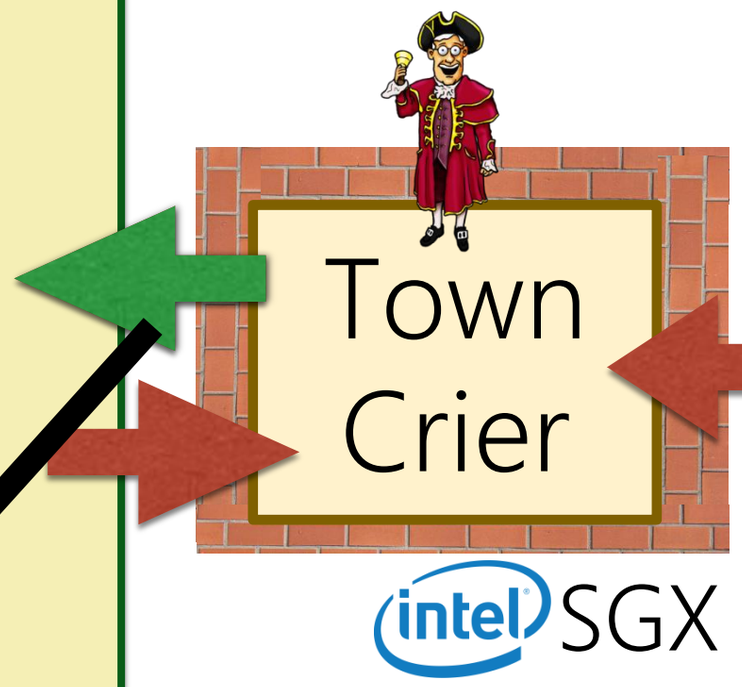




Trusted Website
XYZ.com

- Solution: use attestations to establish secure channel





Trusted Website
XYZ.com

- Creator of Flight Insurance checks

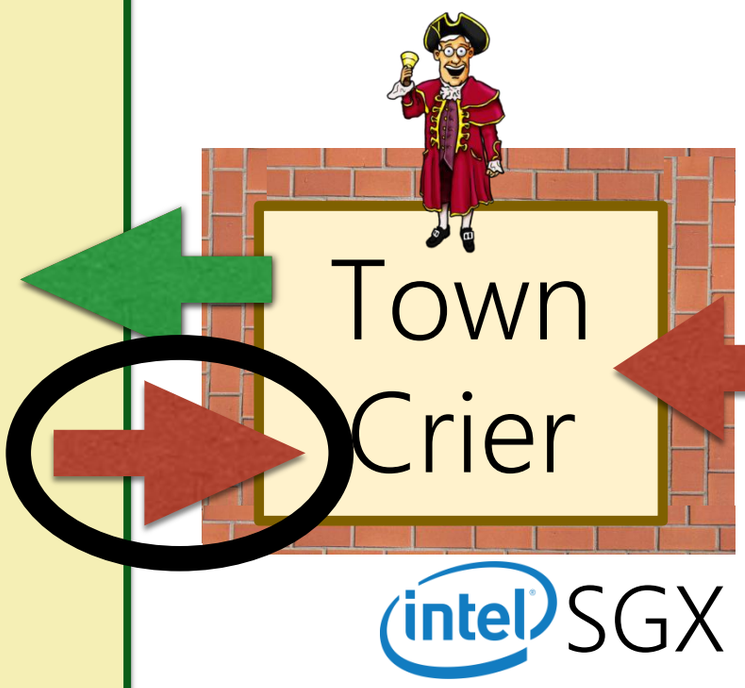
$$\text{att} = \Sigma_{\text{Intel}}[\text{Hash}(\text{TC}) \| \text{PK}_{\text{TC}}]$$

- Contract Flight insurance:
  - Hardwired with PK_{TC}
  - On receiving flight data, checks signature $\Sigma_{\text{SK}_{\text{TC}}}$[flight data]

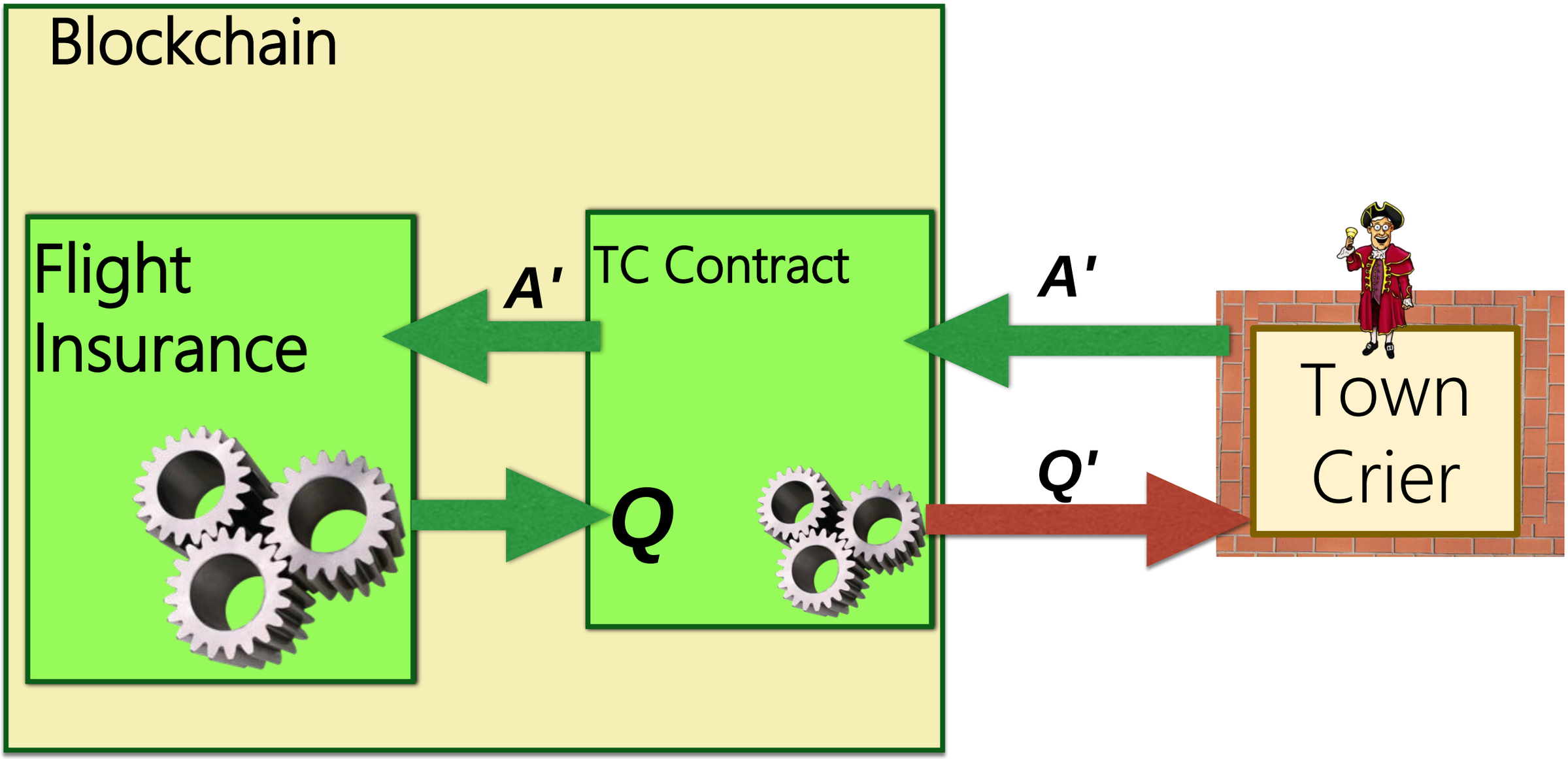
Free!

Blockchain

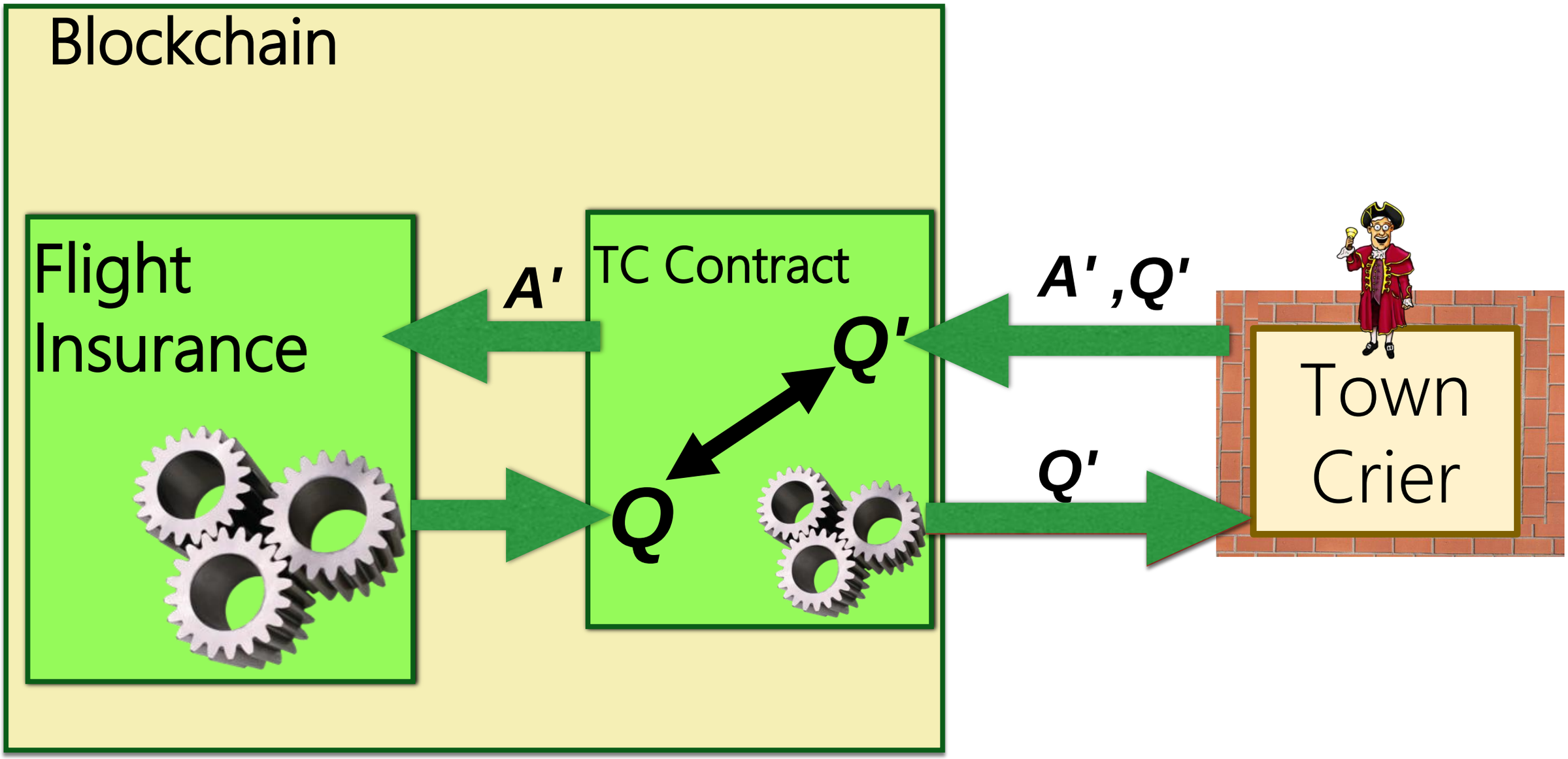
Flight Insurance



Trusted Website
XYZ.com



- Problem: Q' may be corrupted version of Q—or altogether fake!
- Could run blockchain client (Ethereum) in enclave
  - Could verify *Q* from blockchain
  - Would bloat TC code (TCB)!



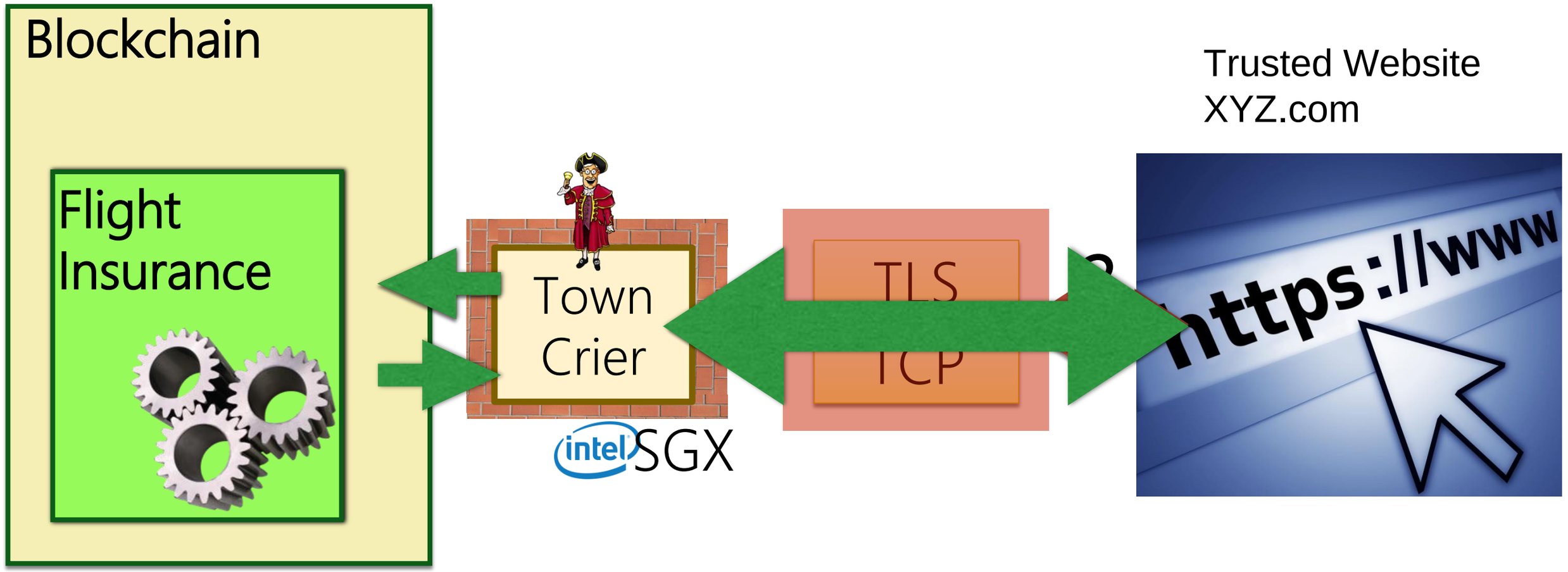
- Our approach: Leverage hybrid trust model
- Potentially *corrupted* query *Q'* processed by TC
- TC digitally signs (*A', Q'*) using SK_{TC}
- TC contract verifies *Q* = *Q'*

Blockchain

Flight Insurance

intel SGX

Trusted Website
XYZ.com



Problem: network is controlled by OS

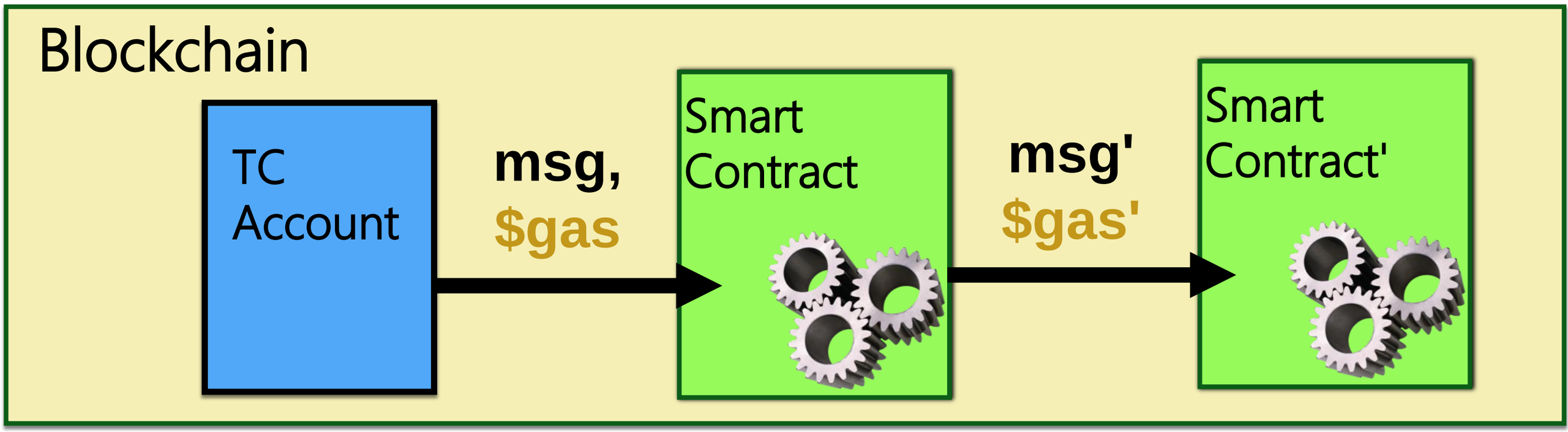
- HTTPS doesn't sign data (MAC is not transferrable).
- TC can not verified data delivered by OS

Solution:

- Put TLS stack *in the enclave*

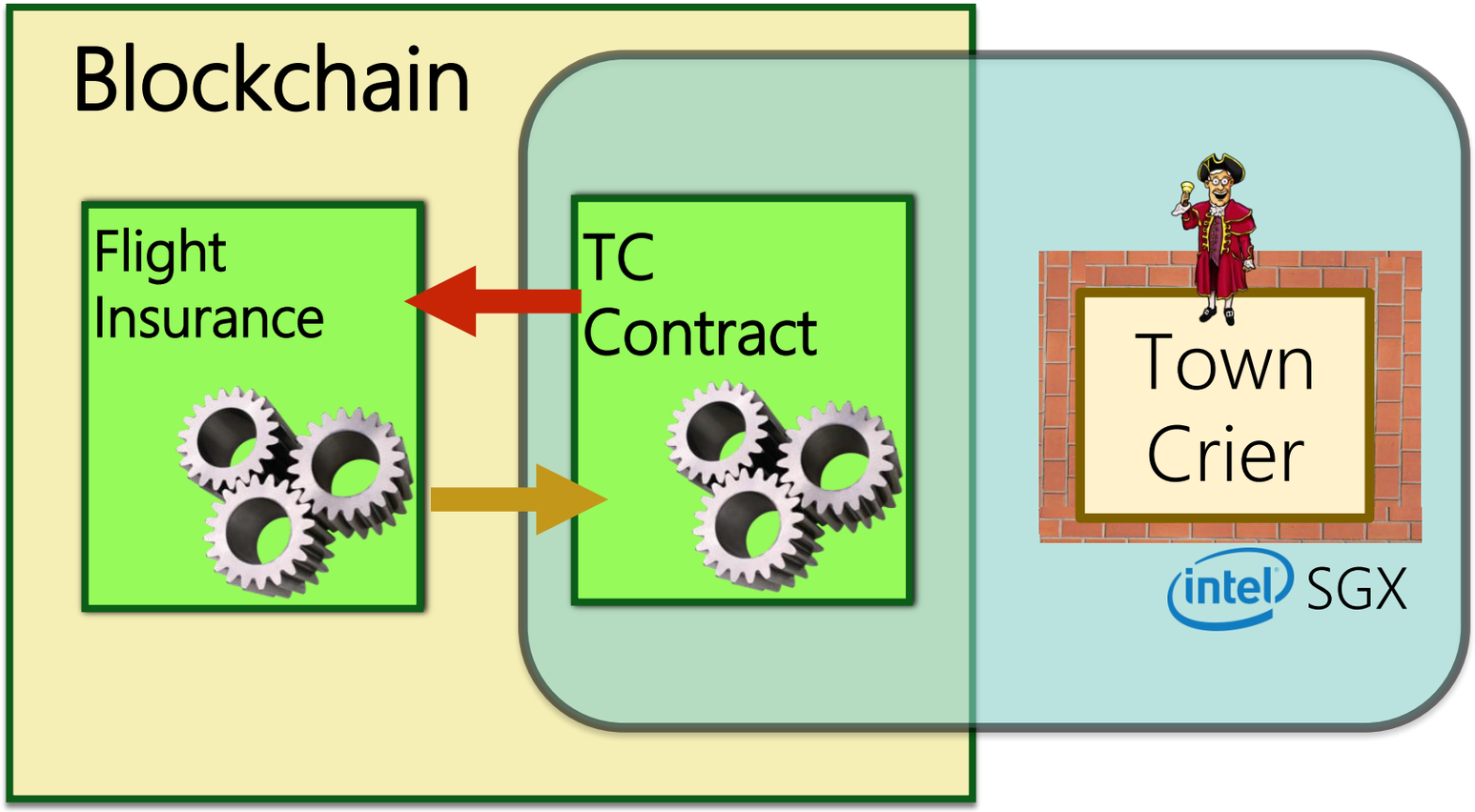
Another problem: Gas depletion

- Recall: Ethereum uses currency called "gas"
  - Prevents DoS
  - Charges fairly for computation costs



Another problem: Gas depletion

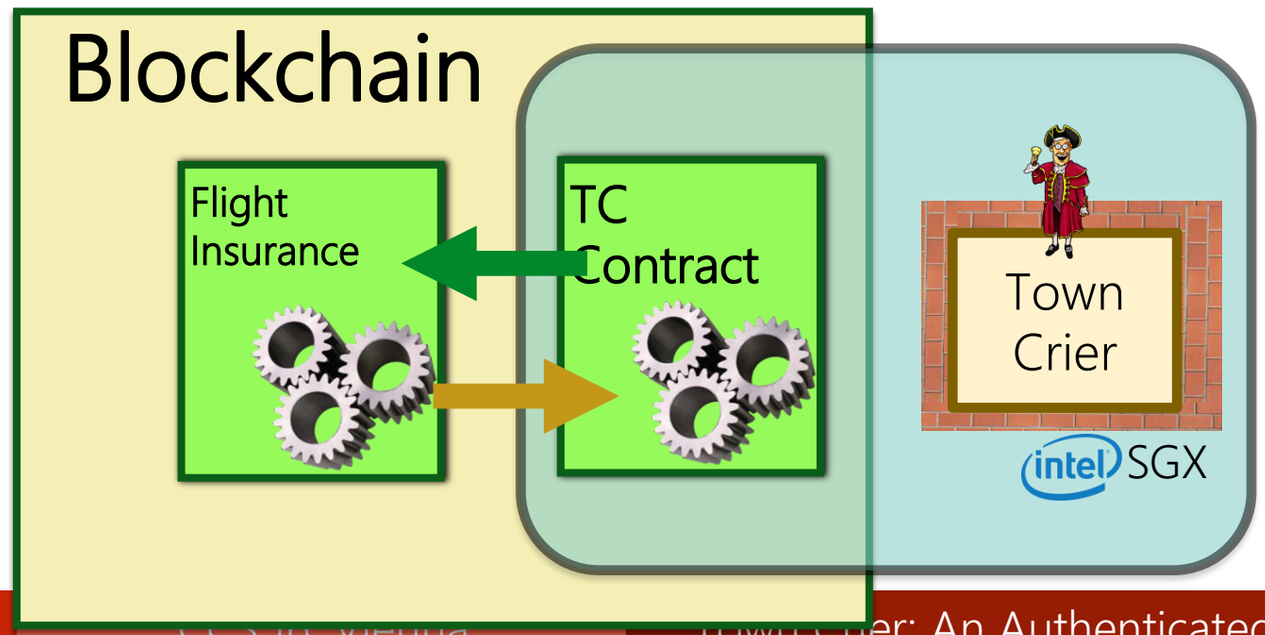
- TC service needs to initiate call to return data *A*
- TC must pay gas cost for this return call… yet avoid *malicious gas-depletion calls from other contracts*



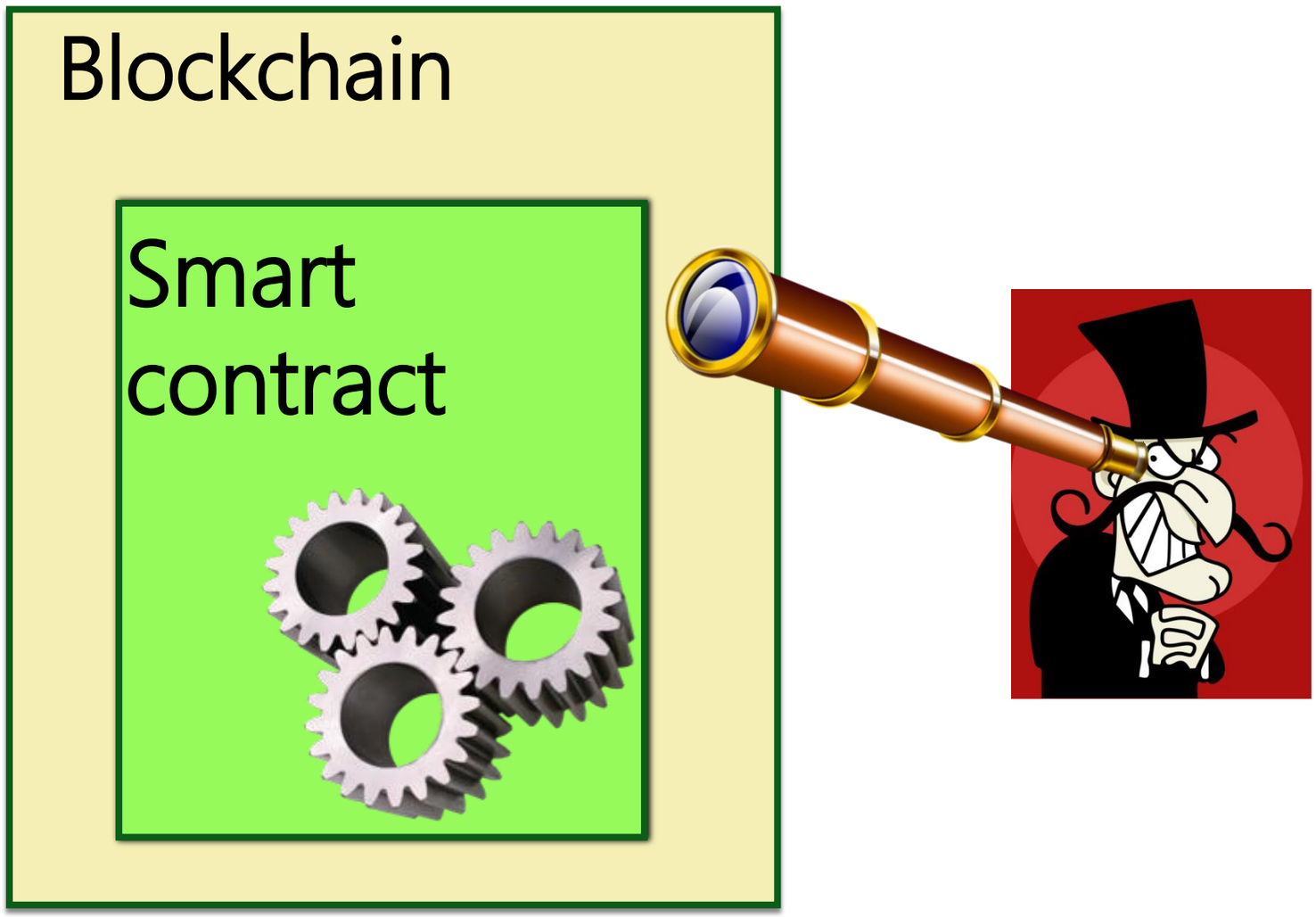
Gas sustainability

- Informally: service never runs out of gas
- Formal definition in paper
- TC requires up-front deposit of gas (ether) from relying contract
- Formal proof of gas sustainability for TC in paper

Gas sustainability is a fundamental and general availability property for compound smart contracts!



Challenge of Confidentiality



How to provide confidentiality for data processed by a smart contract with no confidential state?

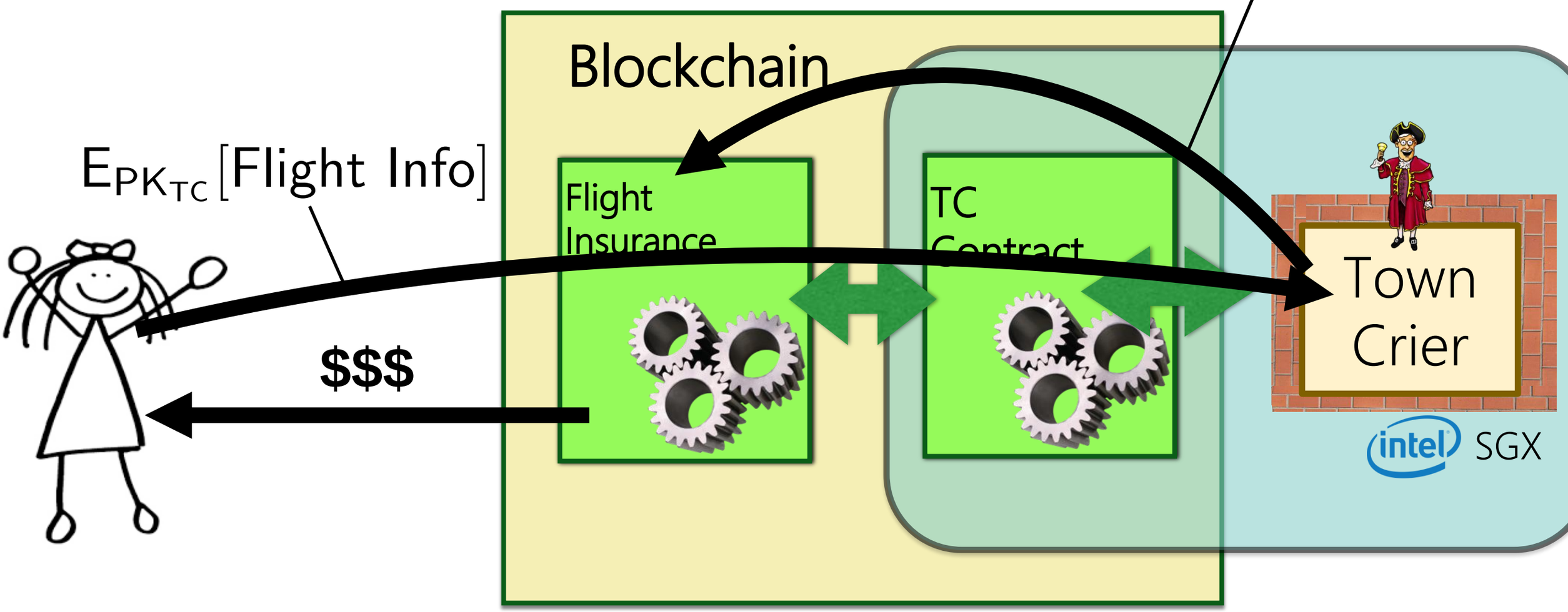
Running example:
Self-enforcing **flight delay insurance**

Flight Insurance

mme a policy for
ght #1234 on 17
May for a policy
price $1.

$100

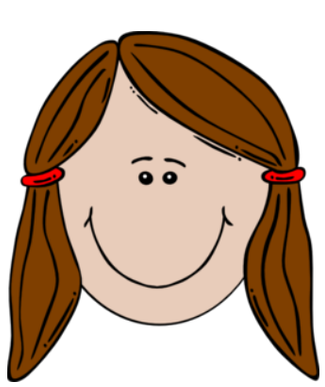
Idea: Leverage enclave confidentiality

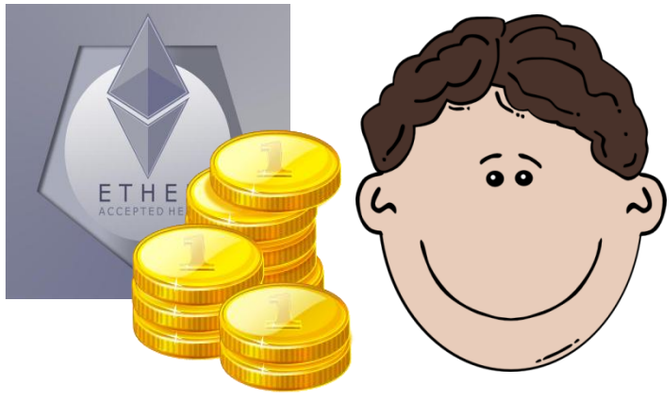


- Private datagram!
- Additional steps needed
  - E.g., delay response / payment to noise timing side-channel

Applications: TC for sale of online goods

- Steam Community Marketplace (SCM): buy and sell items with community members
- Alice wants to sell a game for Ether, but SCM doesn't support.



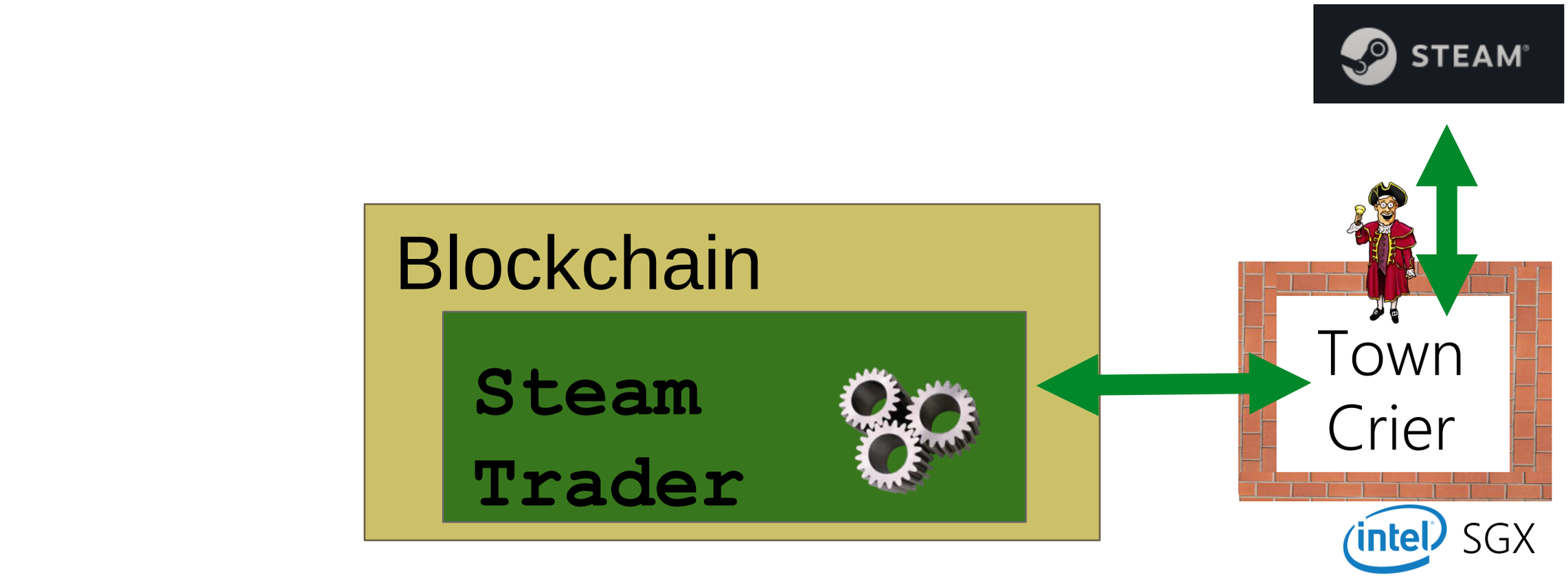


Online game license

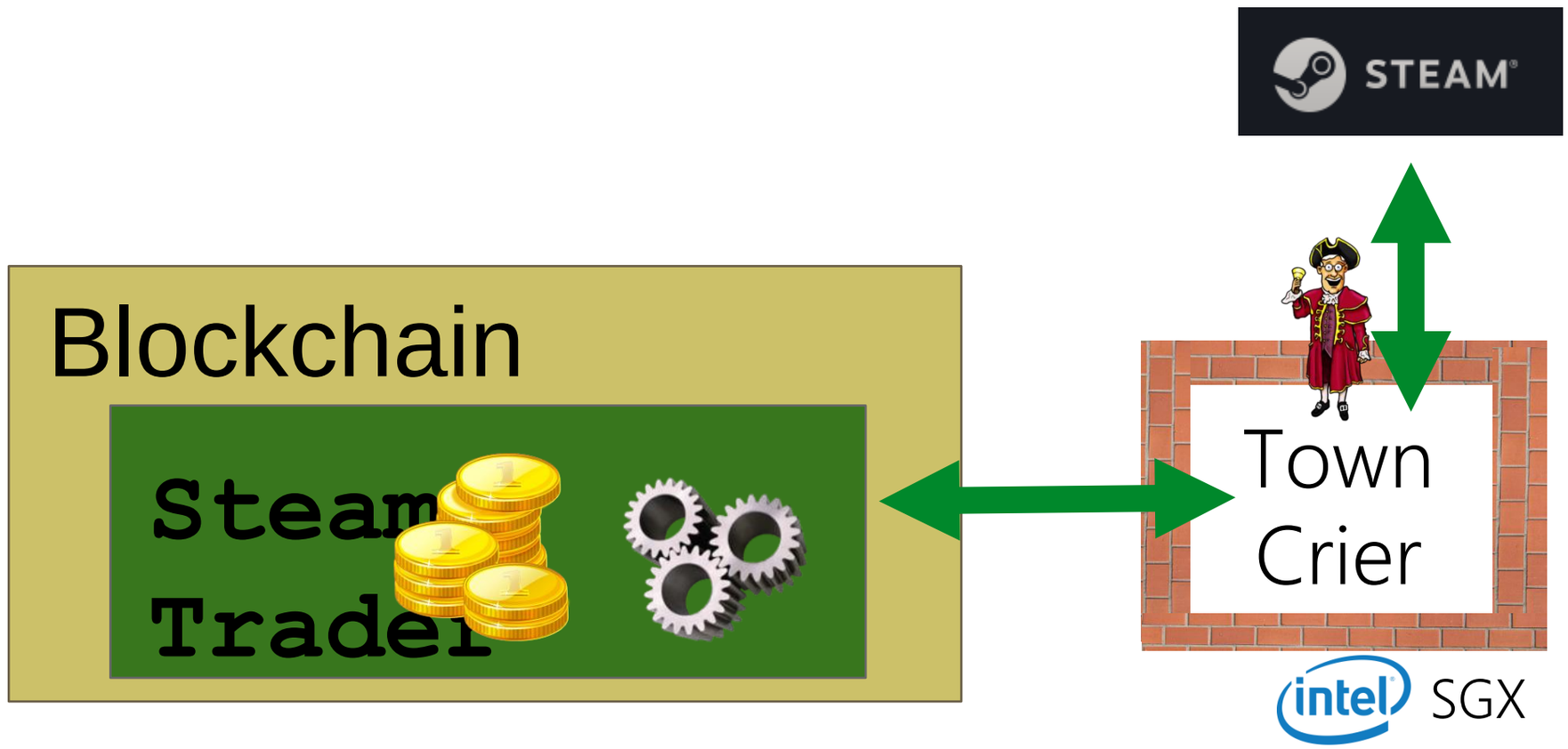
Ether

Community Marketplace

Smart contract for fair exchange

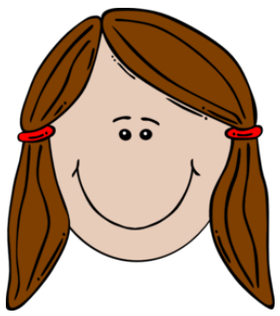


Smart contract for fair exchange

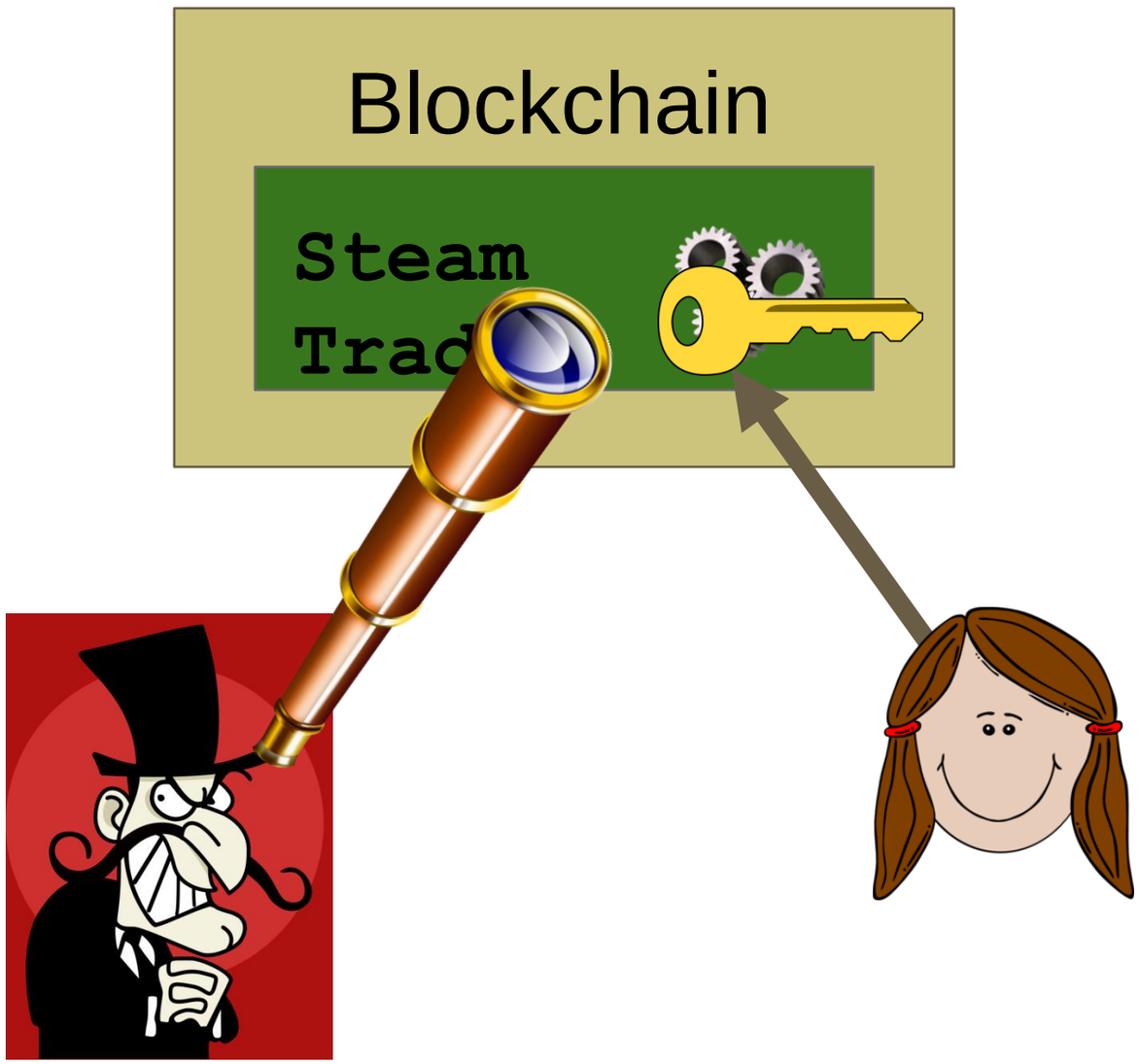


STEAM™ as a gift ($0) through Community market

Smart contract for fair exchange

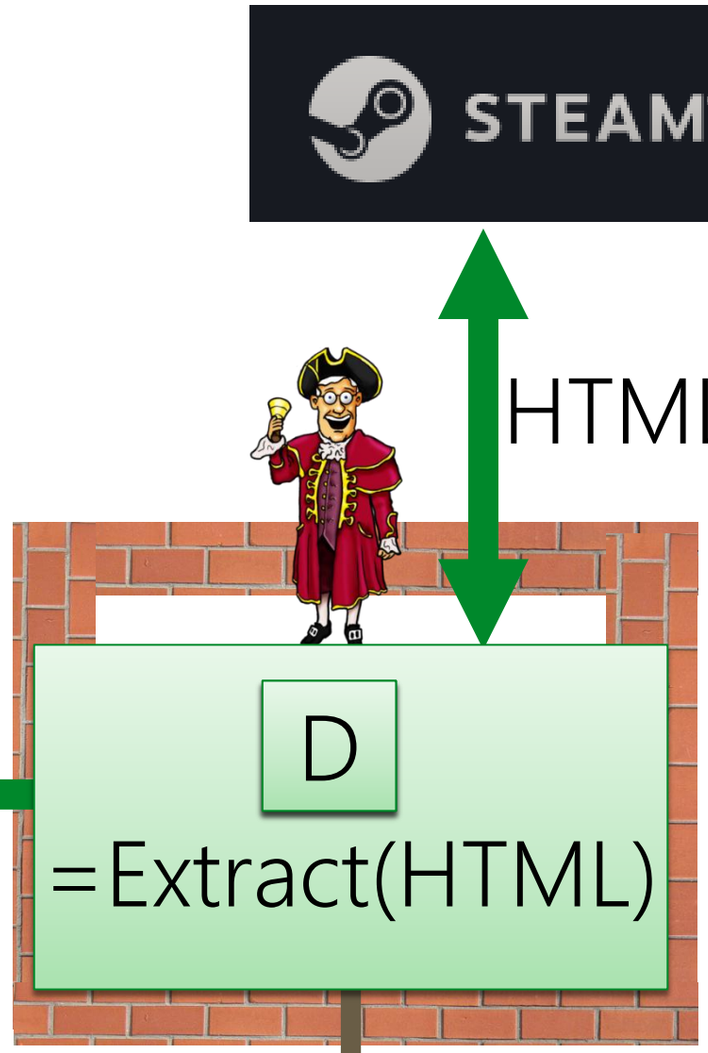


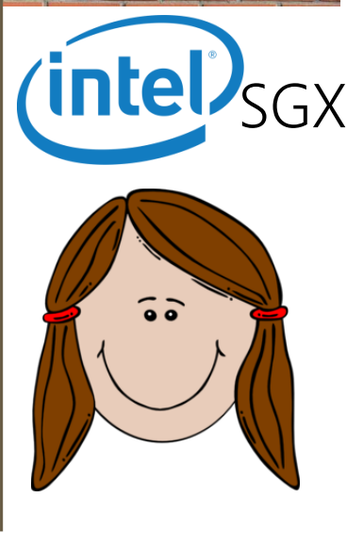
Challenges



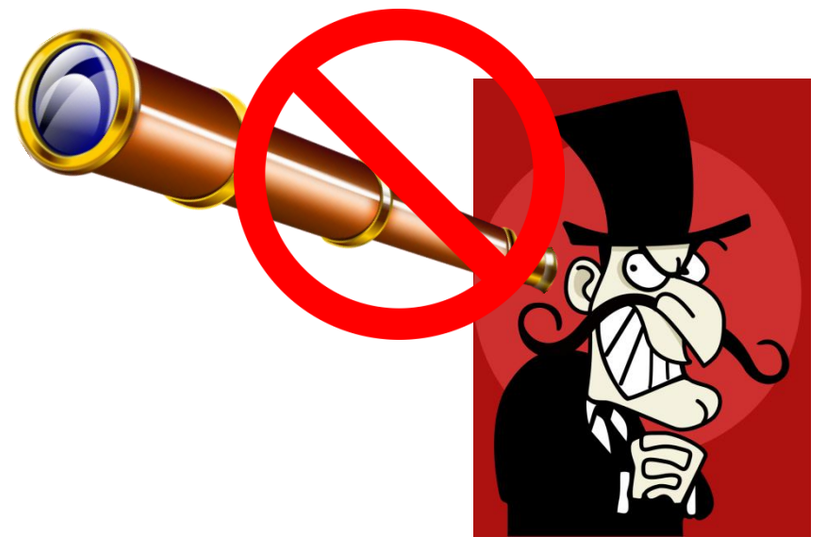
- **Confidentiality**: This requires Alice's (or Bob's) Steam marketplace **credentials**
- **Customization**: Sending raw data to SteamTrader incurs high overhead.

Again, we can leverage SGX



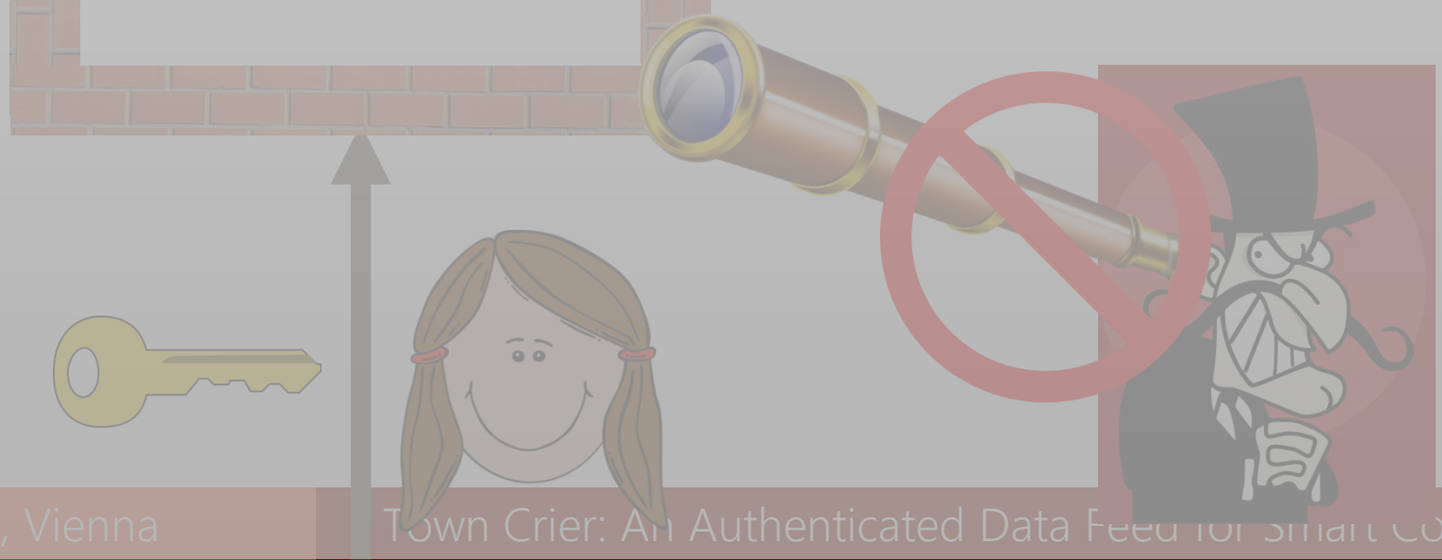


$$E_{\mathrm{PK_{TC}}}[\ \ ]$$

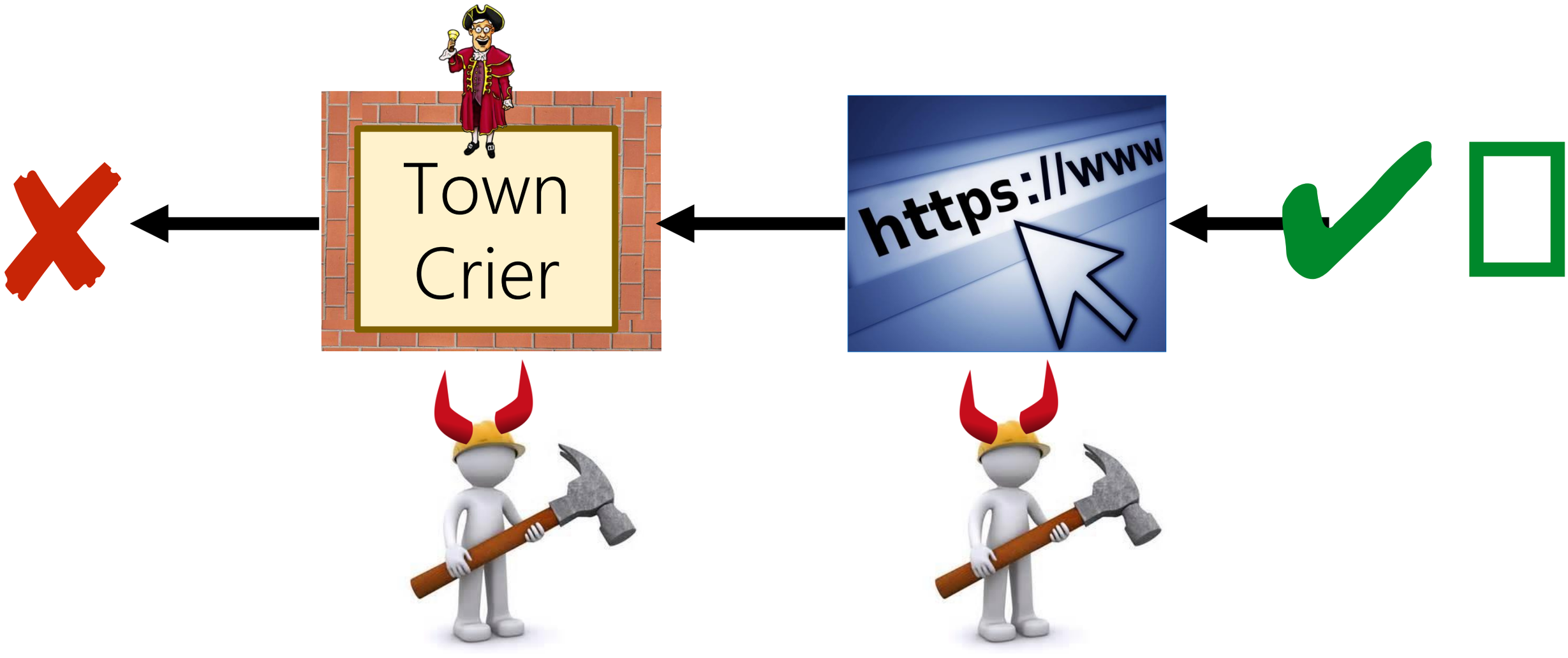


Again, we can leverage enclave confidentiality

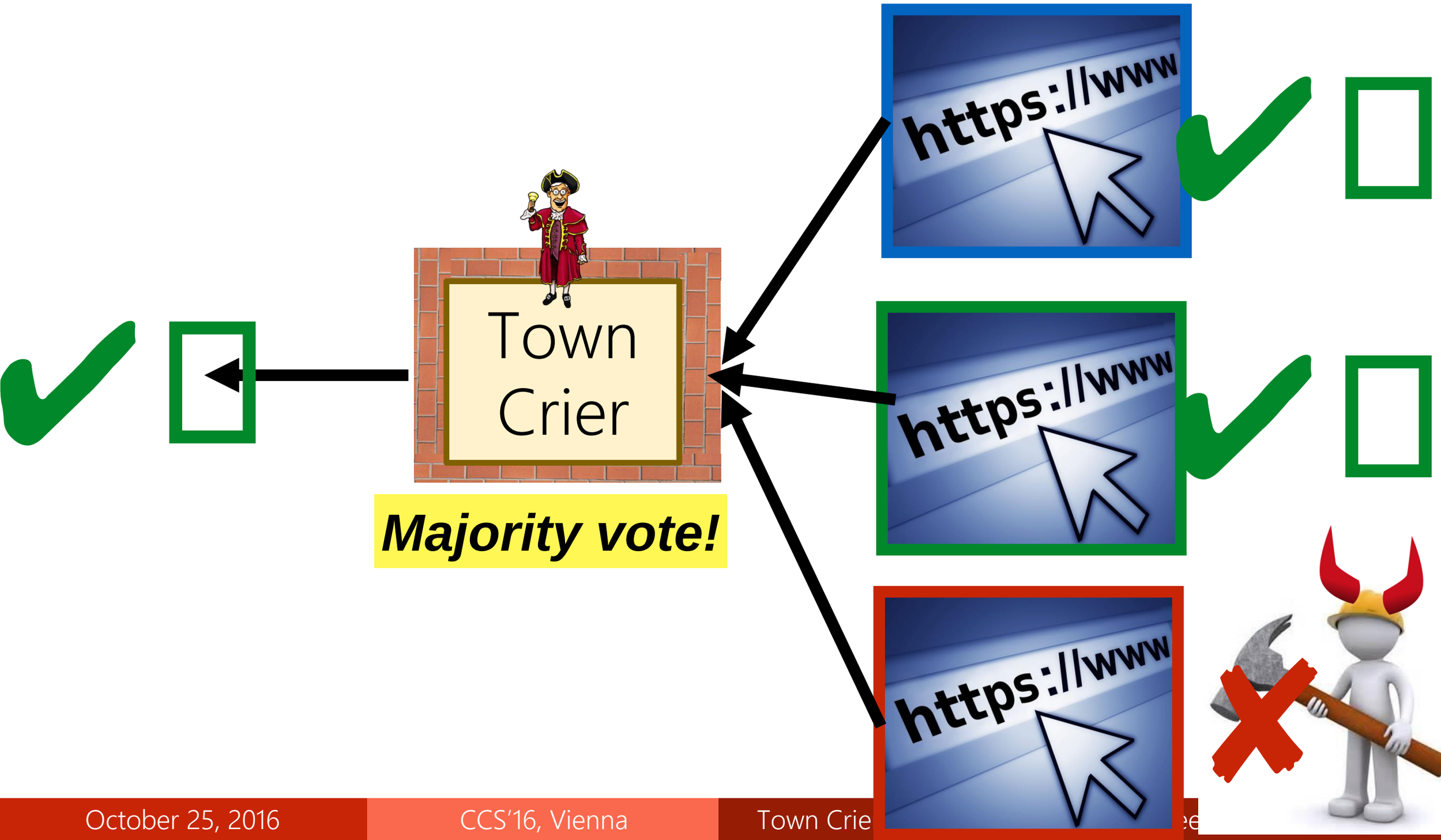
TC enables confidential smart contracts for *nearly any digitally represented asset* ($, cryptocurrency, online accounts, etc.)



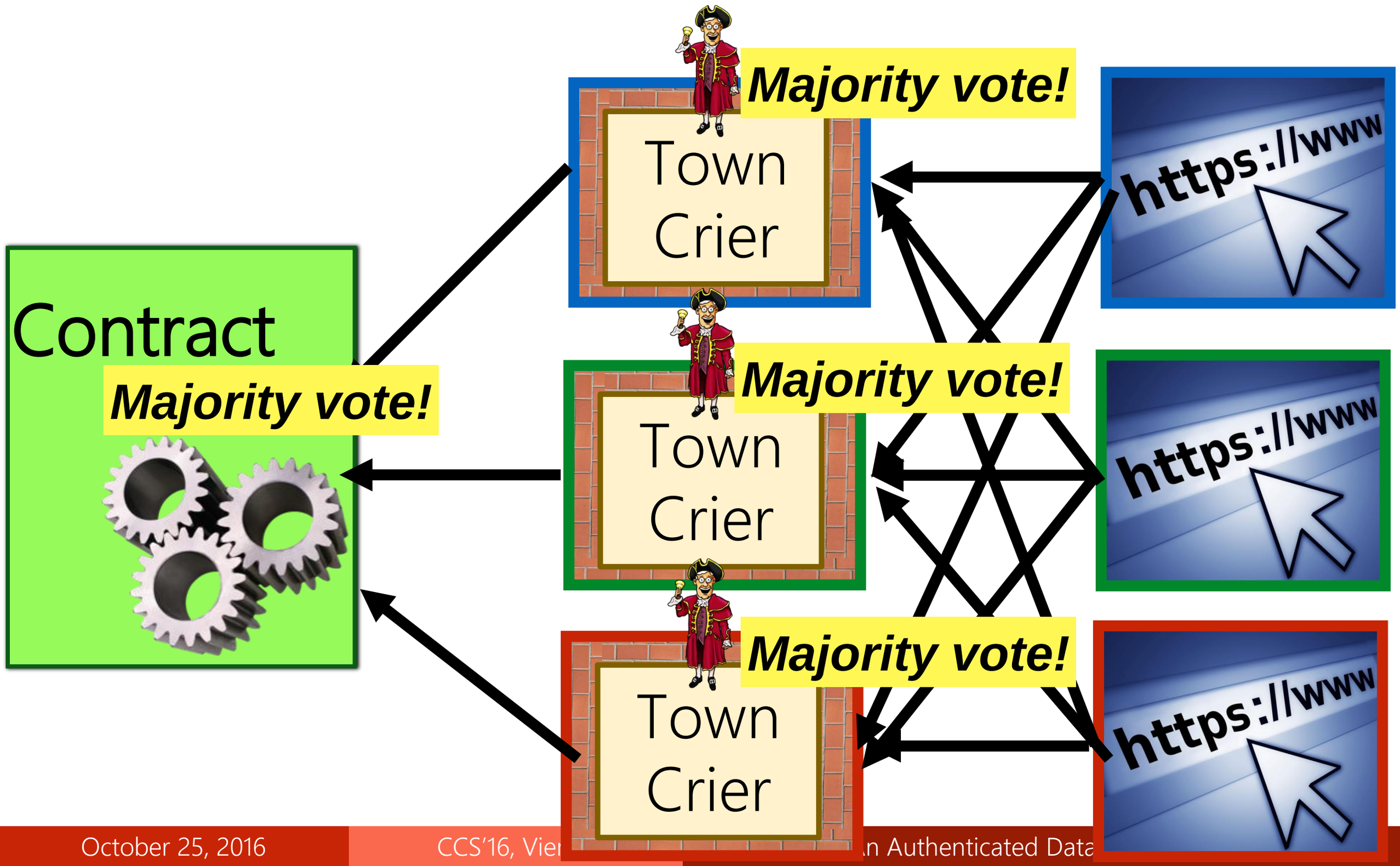
What if node or source is compromised?



We can have redundant sources



+ redundant nodes



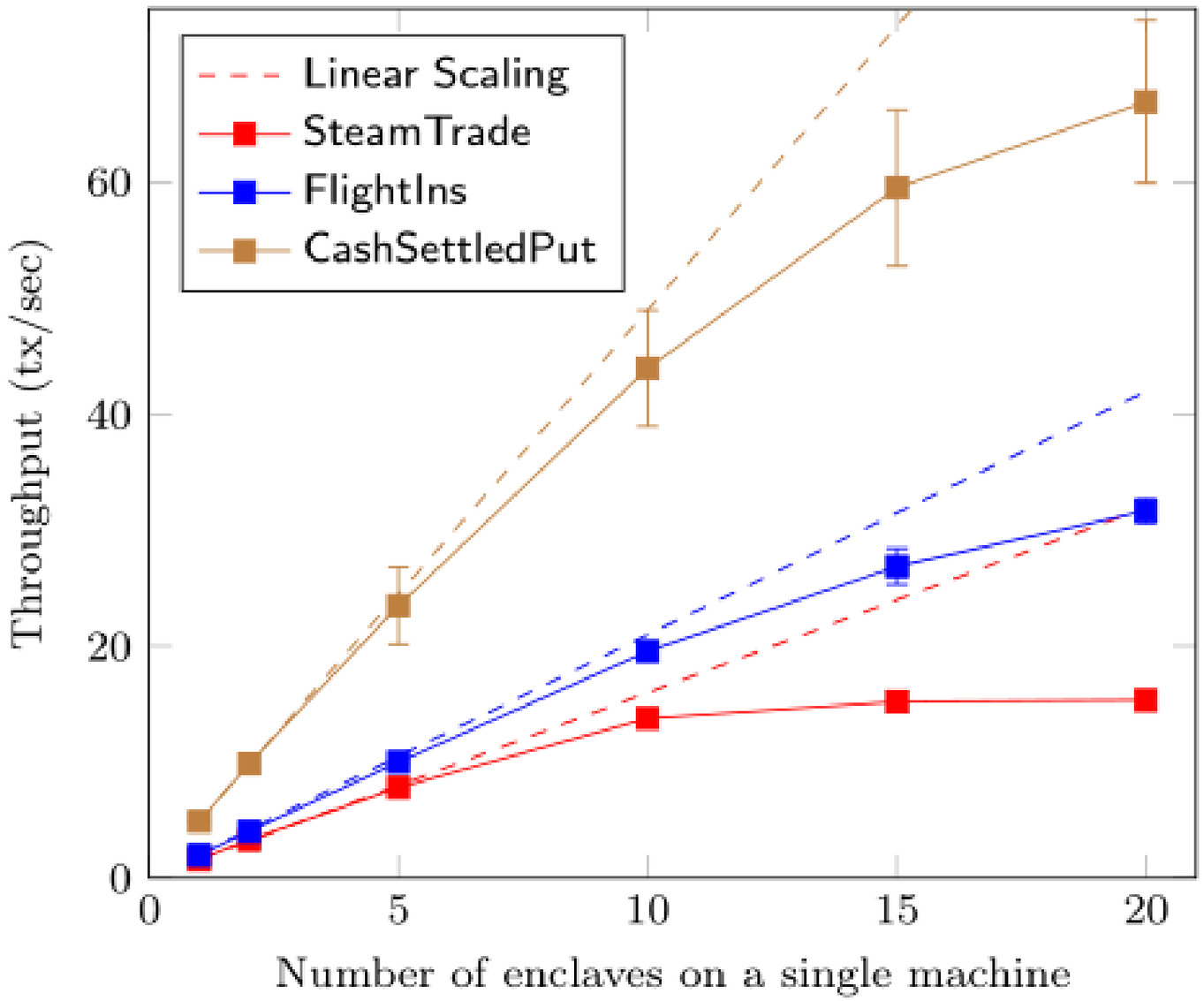


Figure 10: Throughput on a single SGX machine. The x-axis is the number of concurrent enclaves and the y-axis is the number of tx/sec. Dashed lines indicate the ideal scaling for each application, and error bars, the standard deviation. We ran 20 rounds of experiments (each round processing 1000 transactions in parallel).

Town Crier

TC provides authenticated data feeds for smart contracts:

- Has strong security & weak trust assumption
- Preserves confidentiality
- Supports customized data feed

Free version in Ethereum will be launched on **Jan 1, 2017**!