ETHTerakoya

Study on Zero-Knowledge Proofs in Blockchain

Ken Naganuma
Hitachi, Ltd.

Table of Contents

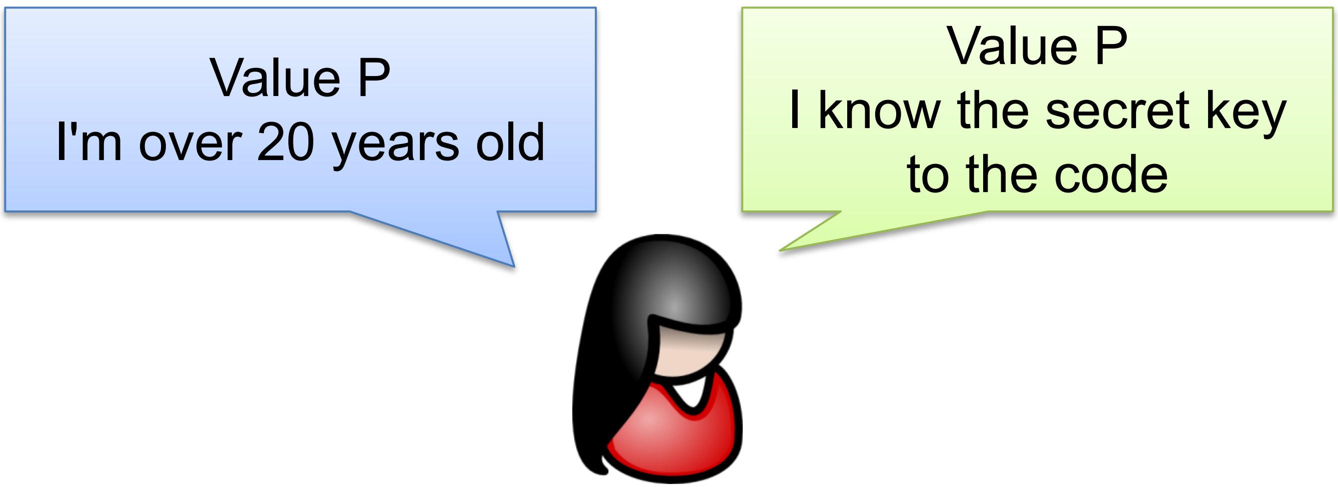
Table of Contents

HITACHI Inspire the Next

What is a Zero-Knowledge Proof?

A zero-knowledge proof is an interactive (or non-interactive) knowledge proof protocol, by which the prover can prove to the verifier that the prover knows a value x, without sharing any information other than the fact that he/she knows the value x.

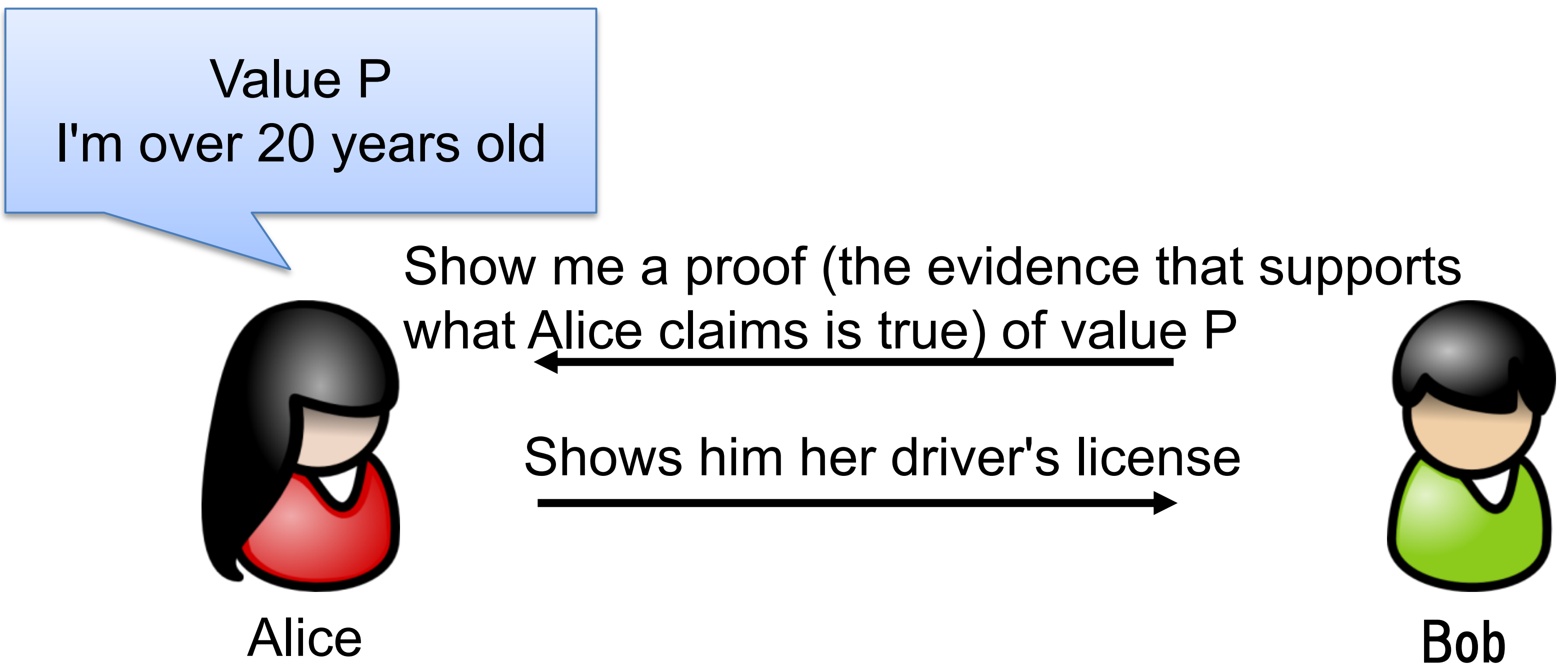
(Reference: Wikipedia)



Alice (the prover) wants to prove to the other party that value P is true.

What is a Zero-Knowledge Proof?

Alice (the prover) wants to prove to Bob (the verifier) that value P is true by using a zero-knowledge proof



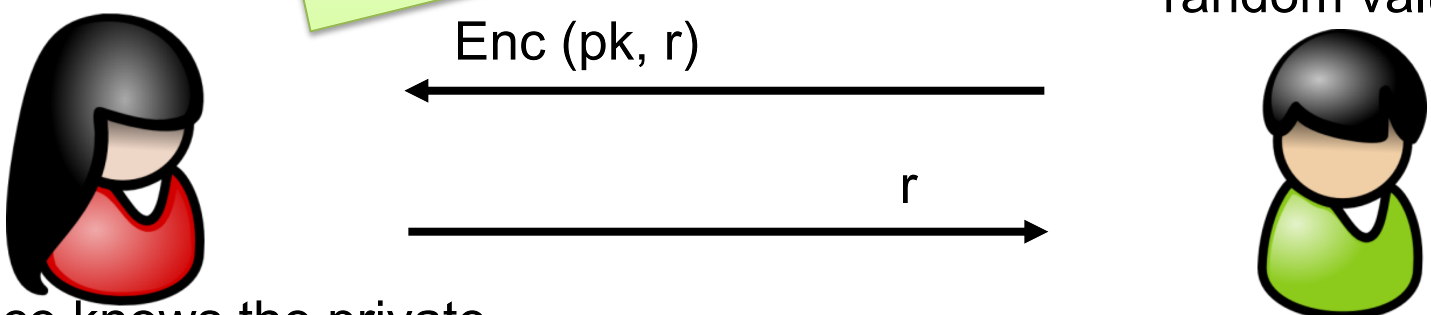
Although Alice (the prover) can prove that value P is true by showing her driver's license, this is not zero knowledge. Some information such as the age, based on her date of birth, is disclosed.
What are the examples of zero-knowledge proofs?

What is a Zero-Knowledge Proof?

Alice (the prover) wants to prove to Bob (the verifier) that value P is true by using a zero-knowledge proof

Value P
I know the private key corresponding to the public key [pk]

Generates a random value [r]



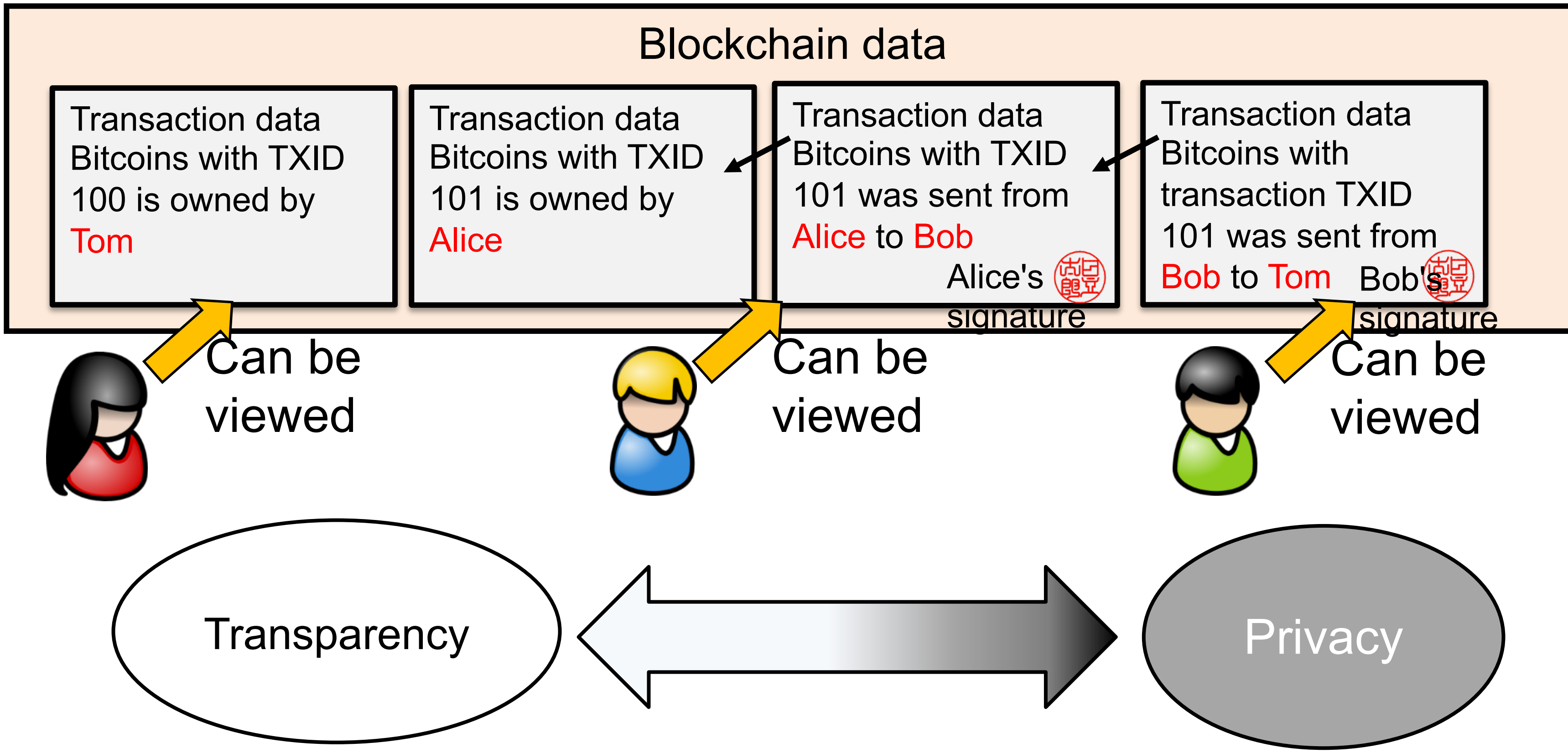
Since Alice knows the private key, she can decrypt Enc (pk, r) to get the plaintext [r]

Since Bob (Verifier) only gets a random value [r] that he generated himself, he doesn't have any knowledge of the private key after the interaction
=> Similar to zero knowledge

Table of Contents

Privacy vs. Transparency

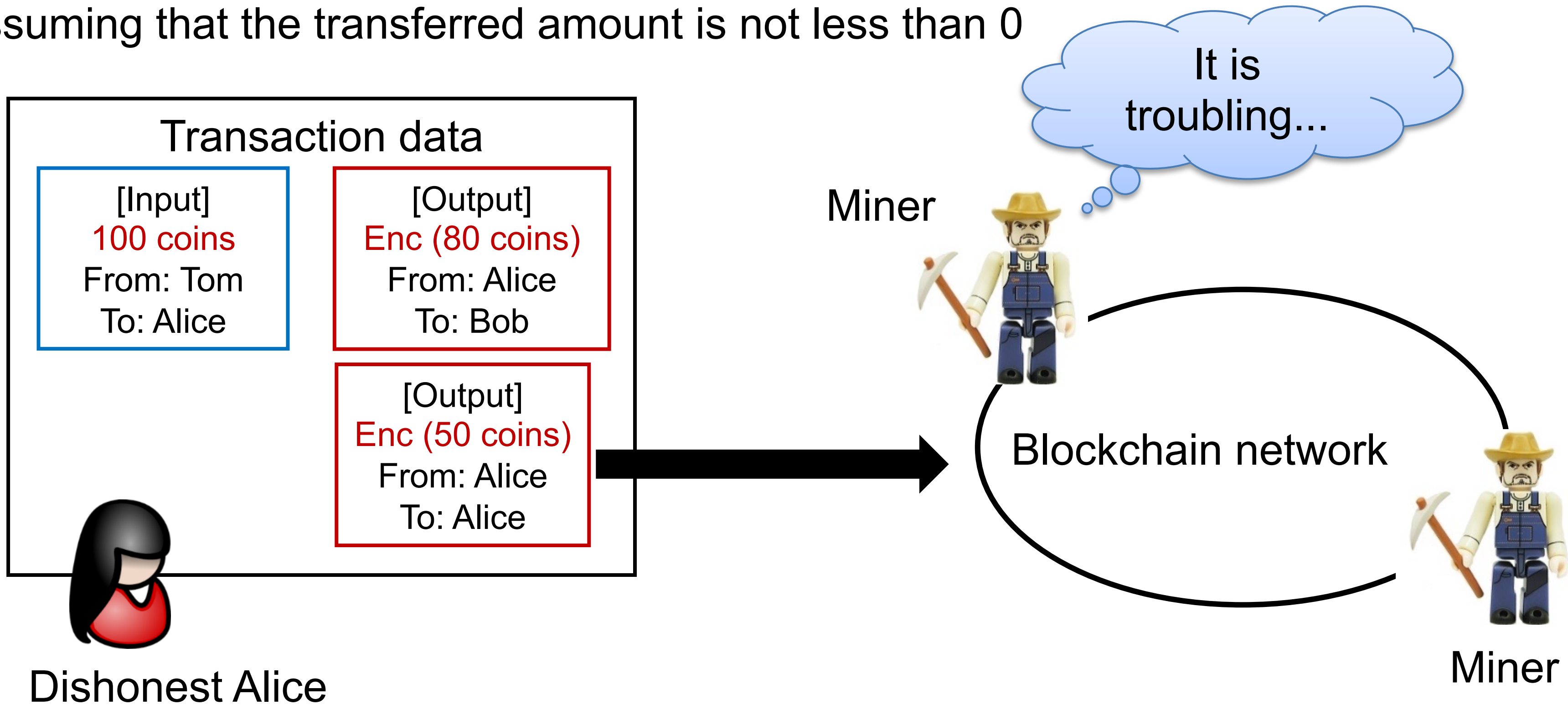
Data in the public blockchain ledger can be viewed by anyone on the network
=> Maximum transparency but no privacy



If the transaction data is encrypted (or hashed), privacy can be maintained but miners cannot verify the validity of the transaction

If the transactions are encrypted...

If the transfer amount was encrypted in the UTXO to protect privacy...
[Total amount of input] = [Total amount of output] (assuming zero commission)
Assuming that the transferred amount is not less than 0



Alice sent 80 coins out of the 100 coins of the unspent transaction to Bob, and sent 20 coins to herself (as change). Since the transfer amount was encrypted, she took advantage and cheated the change as 50 coins.
Because the miner does not have the decryption key, he does not recognize tampering with the amount!

HITACHI
Inspire the Next

The common use of zero-knowledge proofs

Provide validity using a zero-knowledge proof when sensitive information in the transaction is concealed

Zcash illustration

Transaction data
Bitcoins with TXID 100 is owned by Tom

Transaction data
Bitcoins with TX
101

Transaction data

It seems there's no fraud!

Concealed transactions + a zero-knowledge proof

Sensitive information remains concealed but has a zero-knowledge proof Miner

Summary (up to this point)

HITACHI
Inspire the Next

Since the information in the blockchain ledger can be viewed by anyone, there is a privacy concern

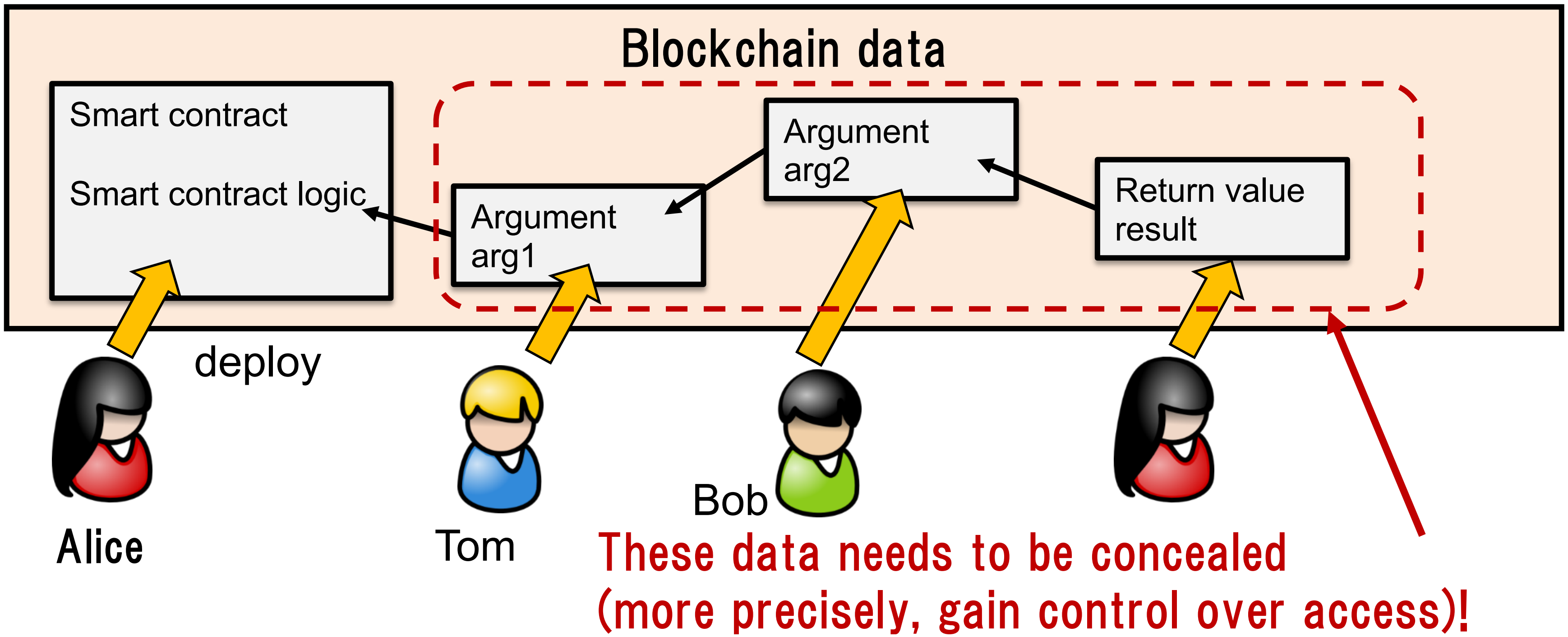
Then we can encrypt the sensitive information

But miners cannot verify the validity of encrypted data

We can use zero-knowledge proofs to prove the validity!

Table of Contents

Illustration of a secure smart contract



- Alice (smart contract owner) deploys a smart contract to the blockchain
- Tom and Bob send arguments for the smart contract
- Alice (or someone else) executes the smart contract based on the argument, and the result is recorded in the blockchain

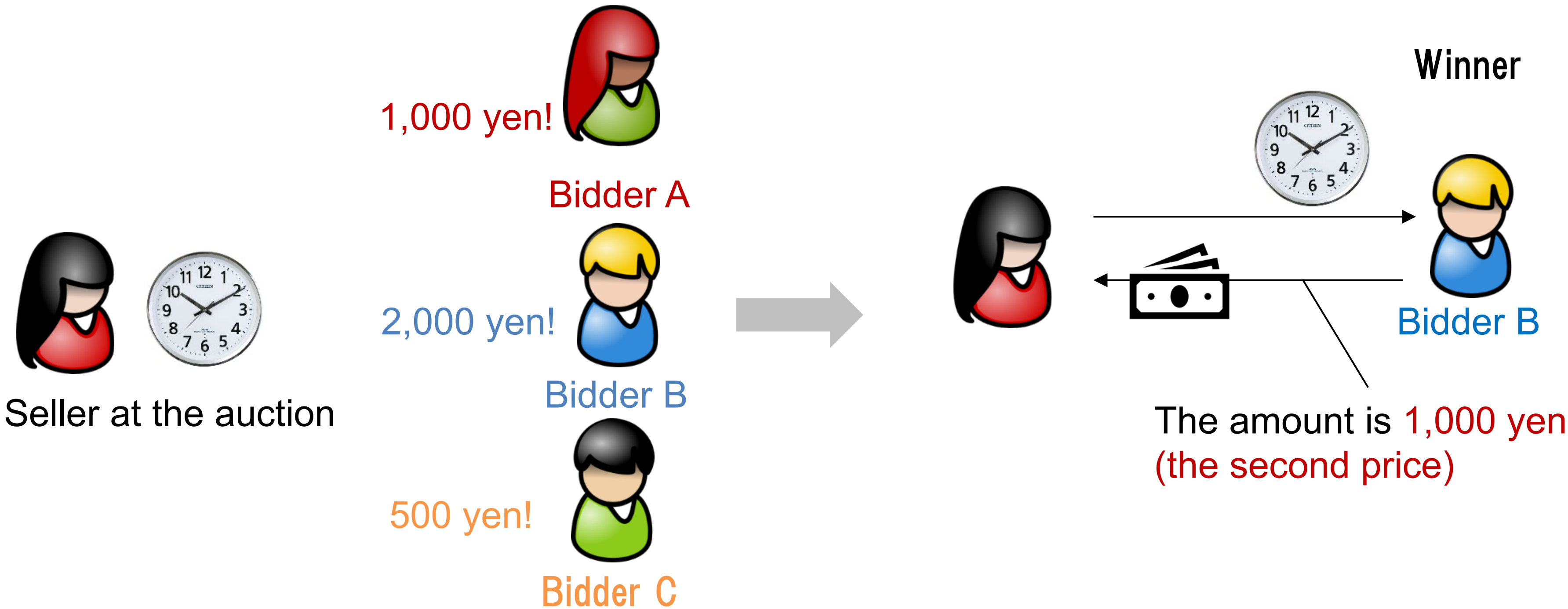
Alice wants to gain proper control over access to these arguments and the result (i.e., limiting access to the arguments to Alice and the party who sent them as well as limiting access to the result to the party involved)

The logic of the smart contract is visible to the public

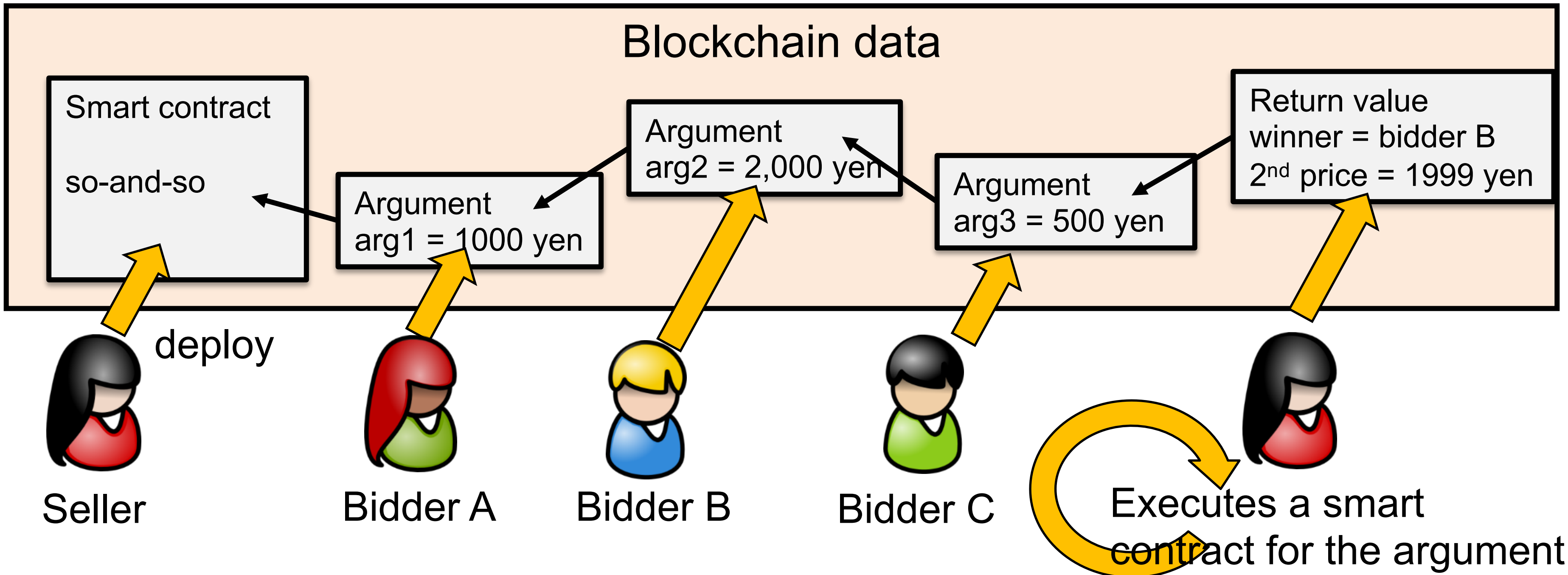
Example 2: The second-price auction

- Each bidder makes only one bid of the highest price that can be paid
- The highest bidder becomes the winner (Bidder B in the figure below)
- The winner pays the second bidding price (1,000 yen bid by Bidder A in the figure below)

• Unlike the first-price auction, the winner does not need to worry about paying too much



Can auctions be held on the blockchain?



[Concerns] If the arguments and the return value are made public...
- It is not a second price auction but an open auction (English auction)
- After Bidder B's bid, the seller takes the bid to enjoys the maximum gain at 1,999 yen (first price)
- All bids and the winner are made public (privacy concerns)
  ⇒ Arguments and the return value need to be encrypted

[Concern] If the arguments and return values are encrypted and disclosed only to the seller...
The seller can falsify the execution result (e.g., 2nd price =1,999 yen)

Logic of the second price auction

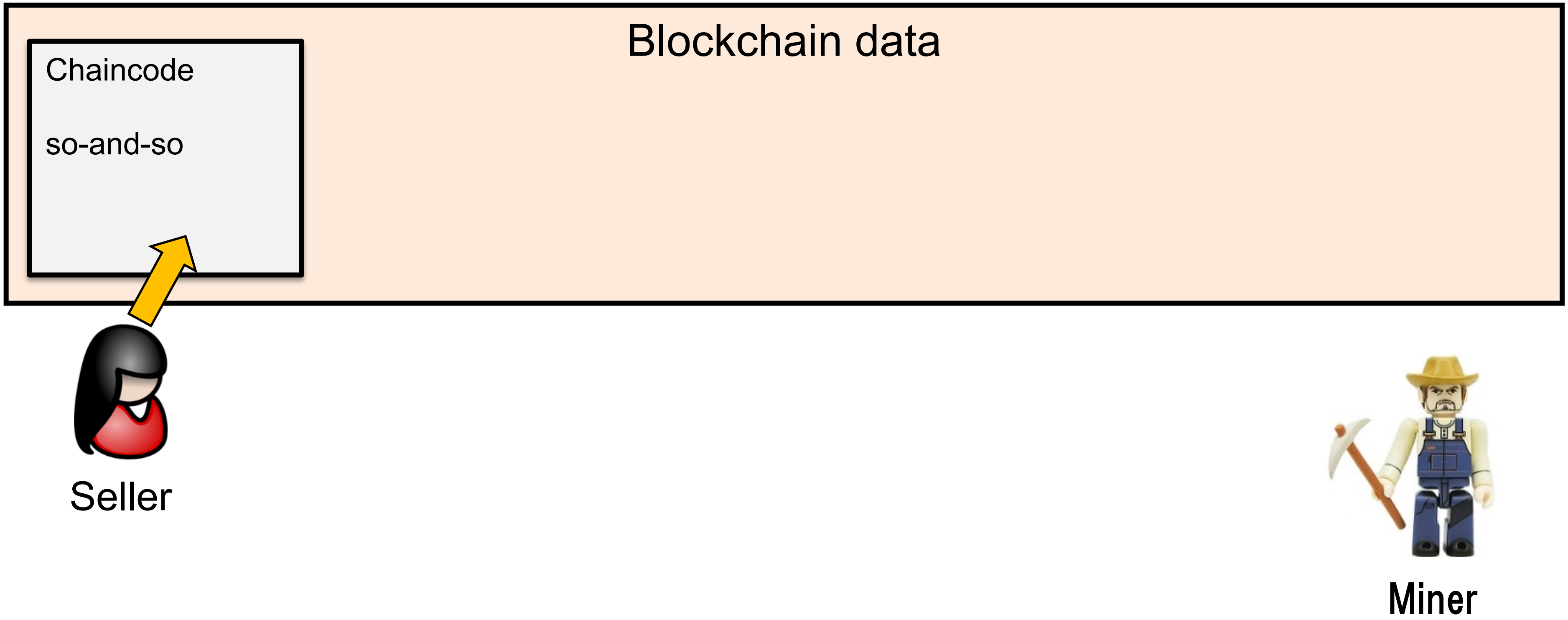
The seller executes the logic after the bidding

```
winner = 0
max = 0
2nd_price = 0
// N is the number of bidders
for i=0;i<N;i++
// Determine the winner
 if bidder[i].price > max
    winner = i
    2nd_price = max
    max = bidder[i].price
return {winner, 2nd_price}
```

* Pseudocode

Since only the seller knows the everyone's bids, no one knows if the seller tamper the bids.

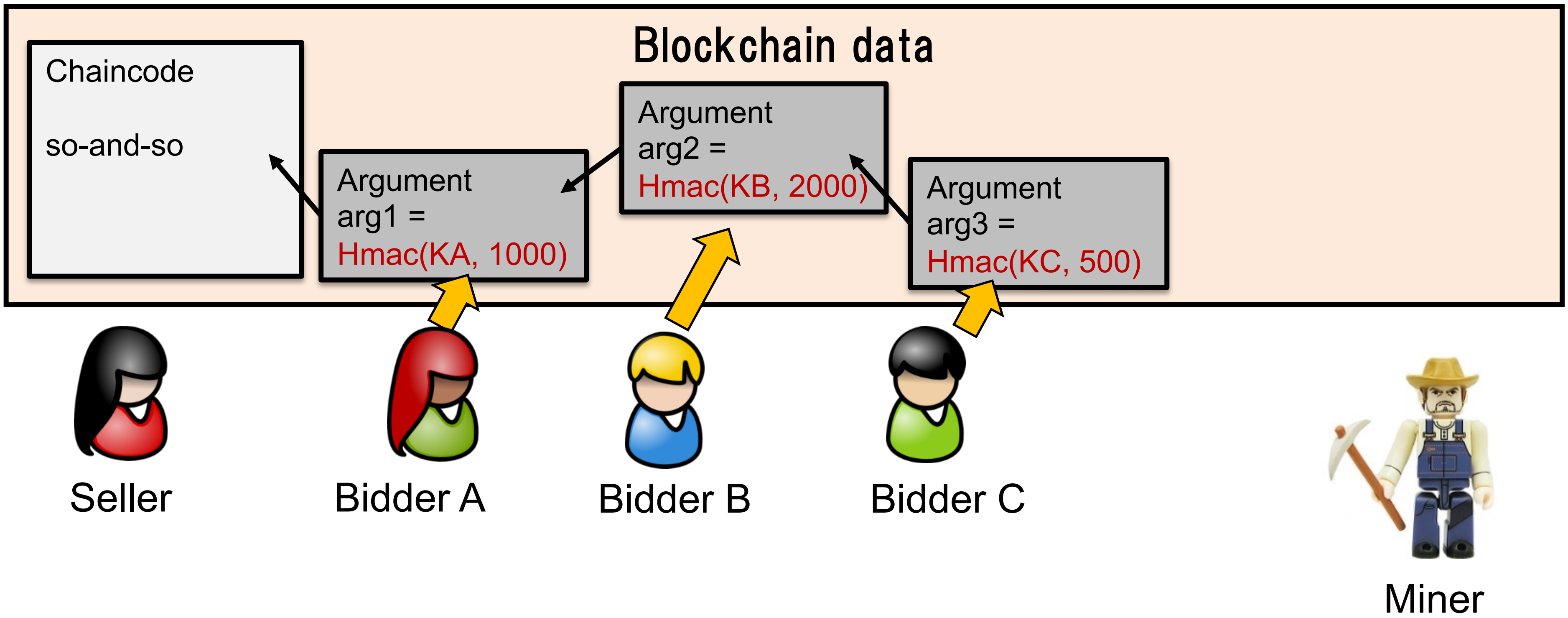
Using zero-knowledge proofs to prevent tampering



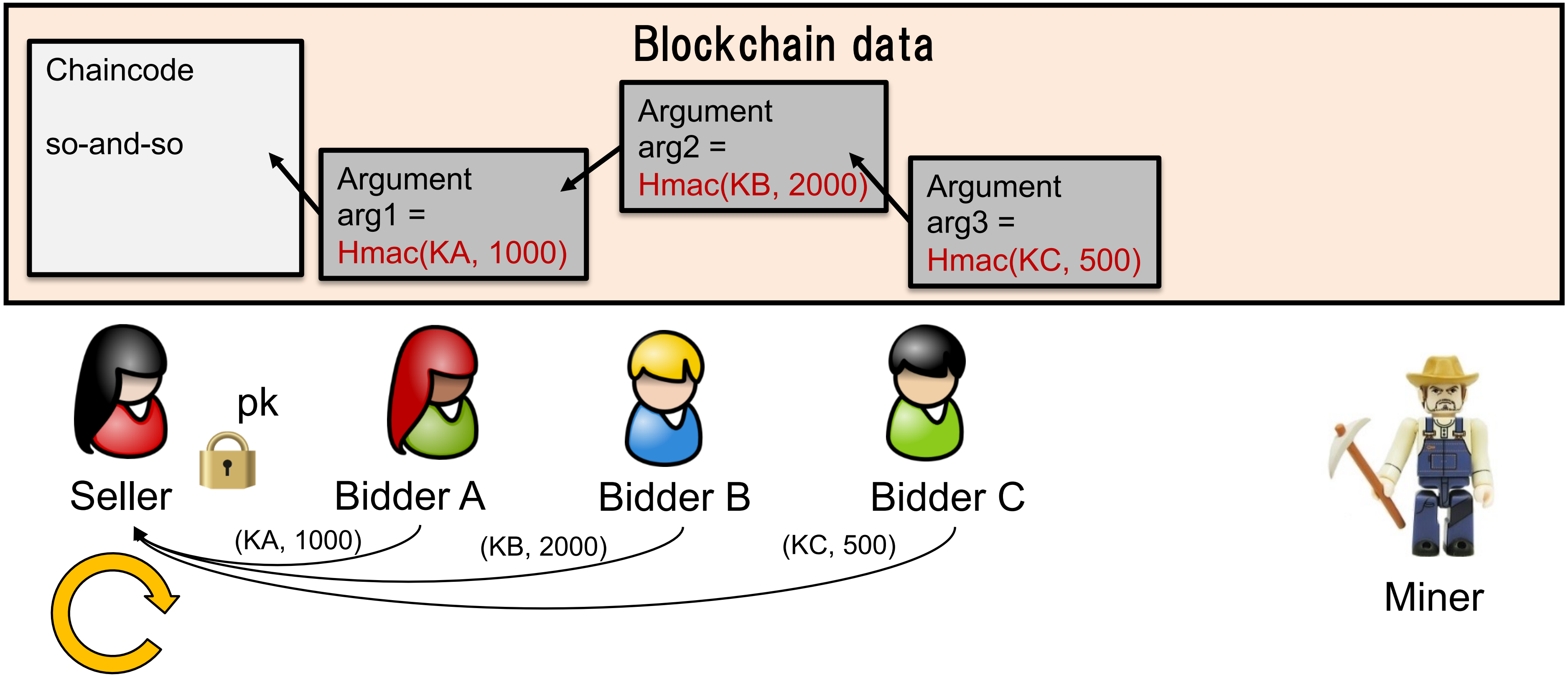
Phase 1, 2 (Initialization and Deployment):
The seller deploys a chaincode for the second price auction

Fabric CA generates zk-SNARK parameters based on the chaincode
- A proving key (pk) for a zero-knowledge proof ⇒ Send to the seller
- A verification key (vk) for a zero-knowledge proof ⇒ Send to the peer

Using zero-knowledge proofs to prevent tampering



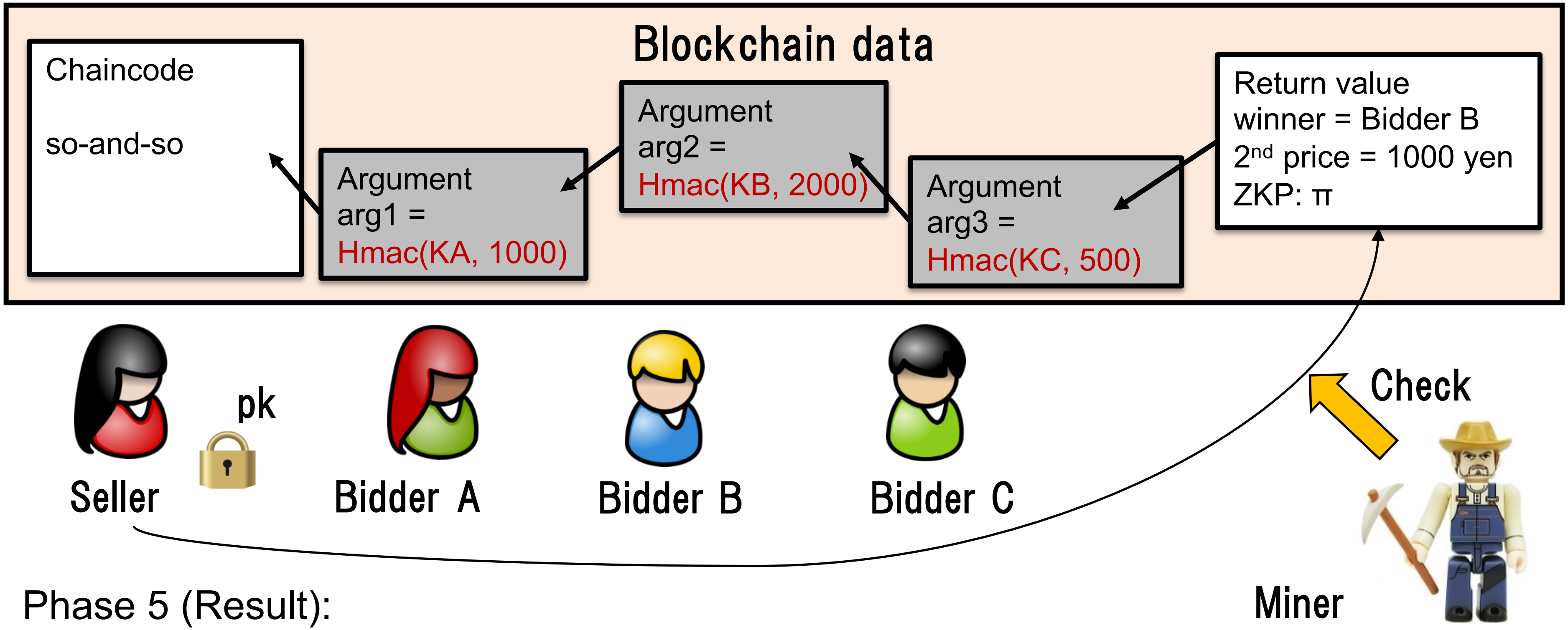
Phase 3 (Bidding):
Each bidder creates a temporary key (KA, KB, KC) and records the HMAC value in the ledger



Phase 4 (Execution): Beware of offline processing!
Each bidder provides a temporary key and a bid to the seller (arguments in the logic)

The seller executes the logic of the second price auction and finds the winner and the second price,
and generates a zero-knowledge proof [π] using a proving key (pk) as a proof that the logic was properly executed

Using zero-knowledge proofs to prevent tampering



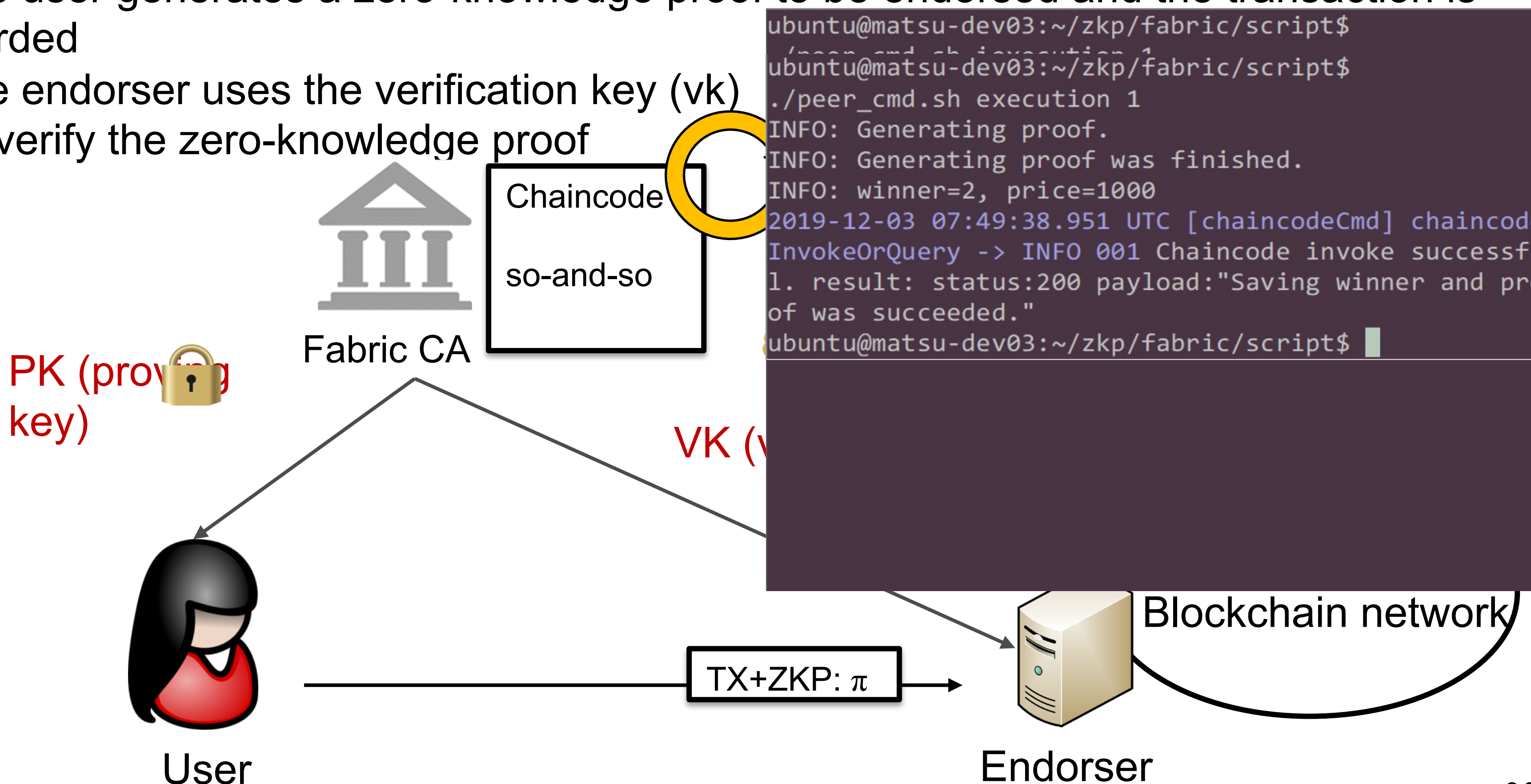
Phase 5 (Result):
The seller records "winner, 2nd price, π" to the ledger (the return value is made public in the current version)

However, the miner will verify that the [logic for the second price] is executed with the [correct arguments (1000, 2000, 500)] (In case of it is invalid, the transaction will not be recorded).
This verification can be done only with the verification key [vk], [π], and the [HMAC value] on the ledger

Table of Contents

Implementation Architecture in Hyperledger Fabric

- The user specifies the contract for which a zero-knowledge proof need to be used
- Fabric CA generates a proving key (pk) and a verification key (vk) for a zero-knowledge proof for the logic of the contract
- The proving key (pk) for the zero-knowledge proof is sent to the user and the verification key (vk) is sent to the endorser
- The user generates a zero-knowledge proof to be endorsed and the transaction is recorded
- The endorser uses the verification key (vk) to verify the zero-knowledge proof



Modified version: Logic for the second price auction

HITACHI
Inspire the Next

Added a function to check the logic for tampering with input amounts (lines 8 to 1

```
1: winner = 0
2: max = 0
3: 2nd_price = 0
4:
5: // N is the number of bidders
6: for i=0;i<N;i++
7:
8: // Check for tampering with input values
9: if bidder[i].hashed_price != hmac(bidder[i].hash_key, bidder[i].price)
10:   abort()
11:
12: // Determine the winner
13:  if bidder[i].price > max
14:    winner = i
15:    2nd_price = max
16:    max = bidder[i].price
17:
18: return {winner, 2nd_price}
```

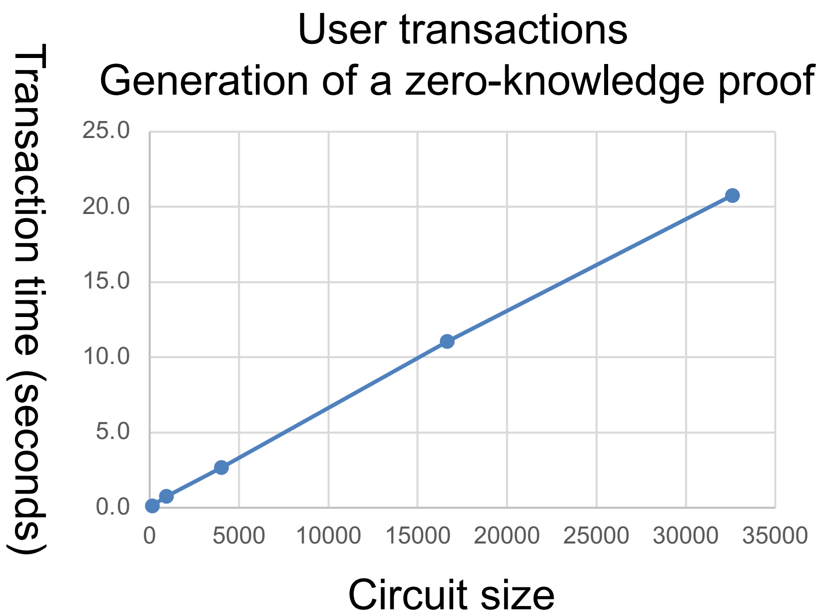
＊ Pseudocode

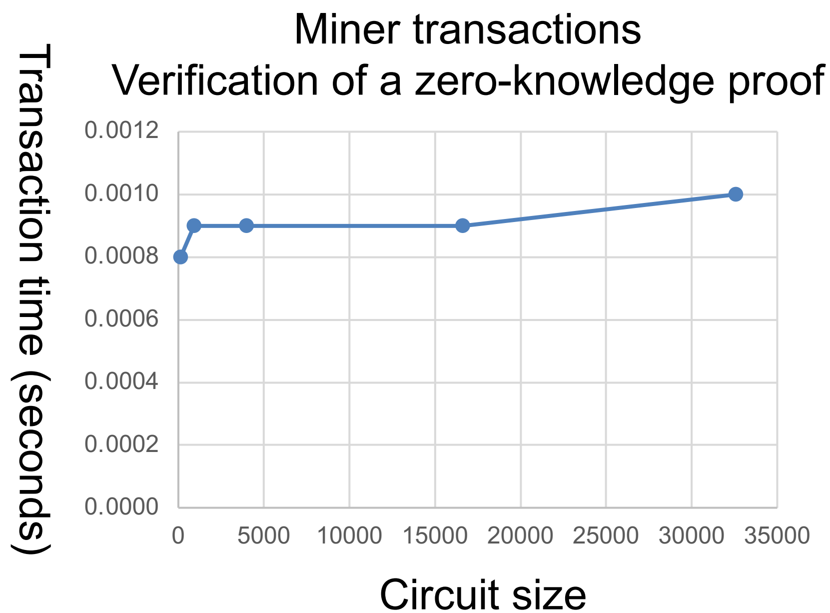
Using only a zero-knowledge proof (ZKP), a verification key, and the public information, the proper execution of the above logic can be verified (verifiable computation)

Table of Contents

HITACHI
Inspire the Next

Performance evaluation experiments





Evaluation method:
Designed the method based on the LWE encryption scheme, considering the resistance to quantum computing
Implemented using some functions of libsnark, the standard library of zk-SNARK
Conducted evaluation experiments for various circuit sizes (smart contract sizes)
Experimental results:

(Computer used: Intel (R) Core (TM) i7-9700K CPU @ 3.60 GHz x8)
- User transaction time (time to generate a zero-knowledge proof) increased in proportion to the circuit size
⇒ Computation of evaluating the hash function requires 300 GB of memory and takes 210 sec
- Miner transaction time (time to verify a zero-knowledge proof) was about 1 msec, regardless of the circuit size

HITACHI

Inspire the Next