网络安全技术

刘振

上海交通大学 计算机科学与工程系

电信群楼3-509

liuzhen@sjtu.edu.cn

Introduction to Bitcoin

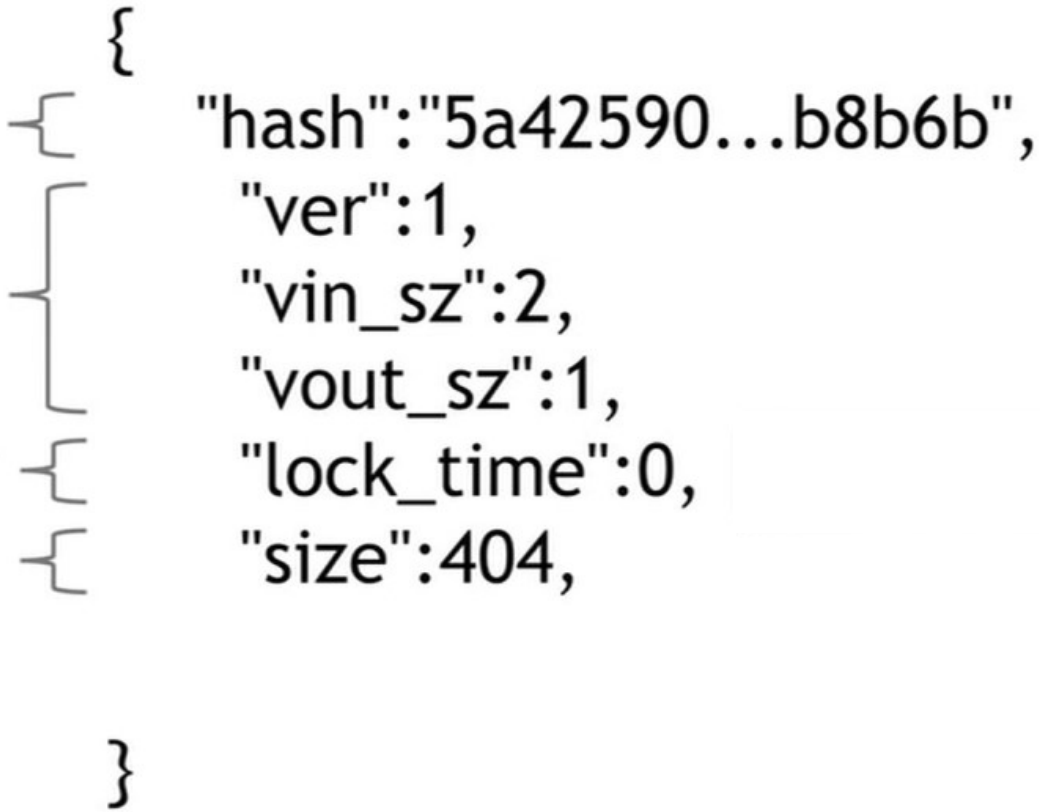
2. Transaction in Bitcoin

- **A Transaction**

2. Transaction in Bitcoin

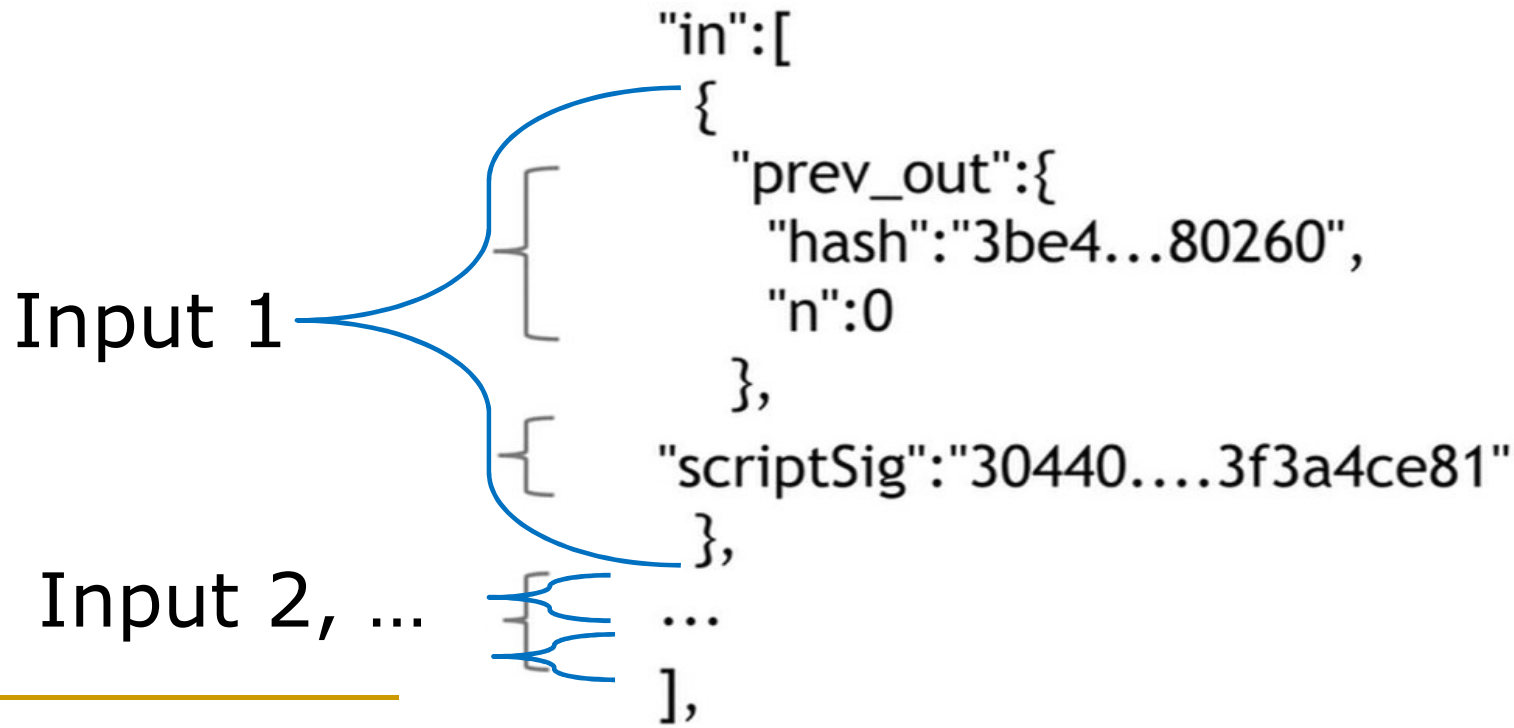
- **A Transaction**
  - **Meta-data**

Transaction hash/ID



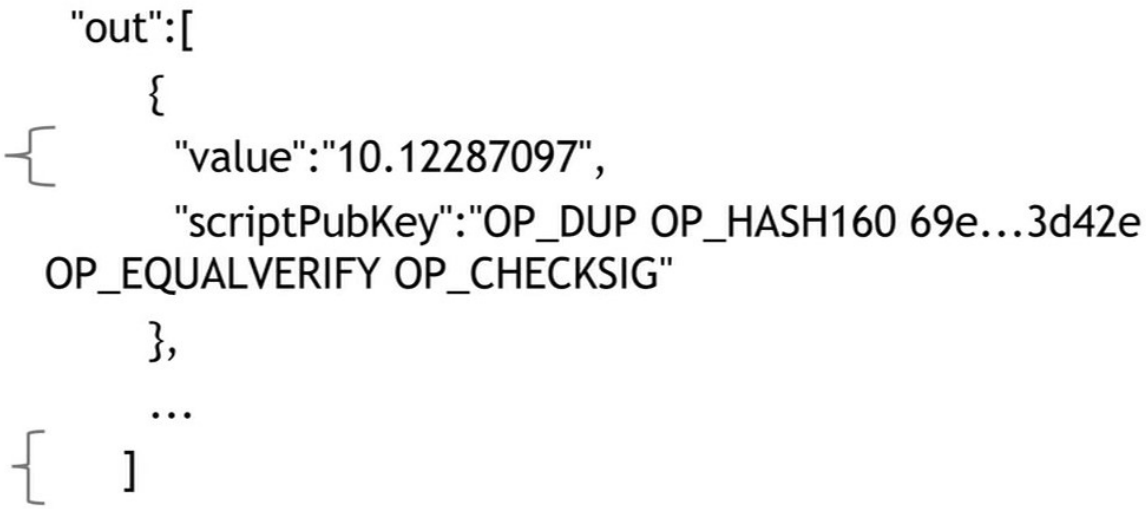
2. Transaction in Bitcoin

- **A Transaction**
  - **INPUTS：**
    - **Each input includes :**
    - **(1) a reference to some output of some previous transaction;**
    - **(2) A proof that the spending is authorized.**



2. Transaction in Bitcoin

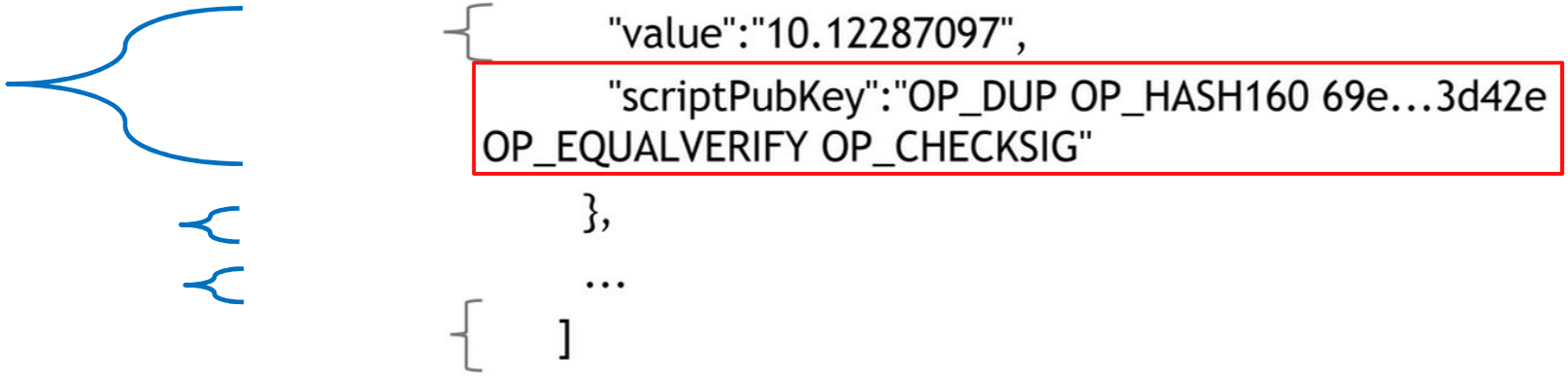
- **A Transaction**
  - Output
    - **Each output includes： （1）a value （2）an address**



A transaction is valid, if
1. The total value of outputs is less than that of the input coins（the difference will be the transaction fee）
2. The public key and signature provided in each input is verified valid
3. The transaction output (TXO) referenced by each input is not referenced by other previous transactions

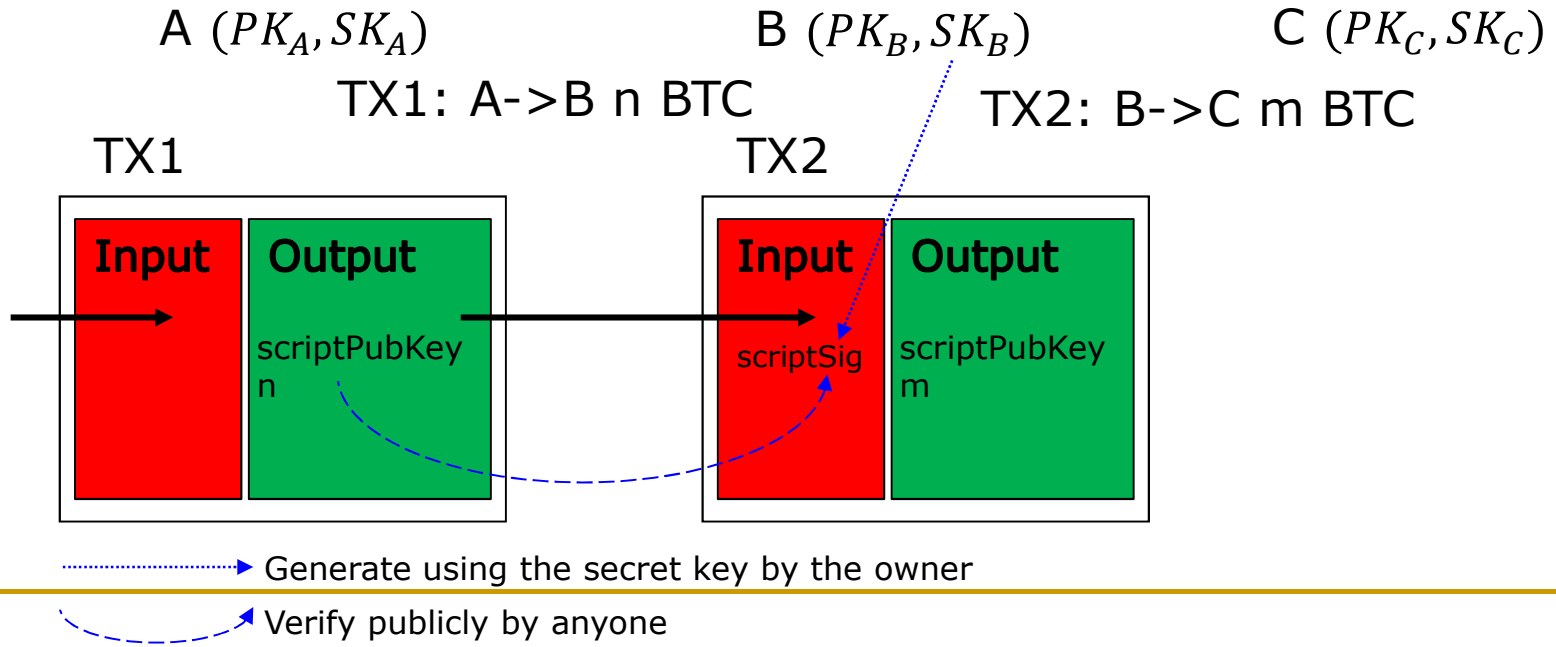
2. Transaction in Bitcoin

- **Script of Bitcoin**
  - The address in TXO is not a hash value of a public key, instead, it is a script source code including operators and operands：scriptPubKey
  - Furthermore, the signature and public key in each input can also be regarded as script source code, including only operands：ScriptSig



2. Transaction in Bitcoin

- **Script of Bitcoin**
  - The address in TXO is not a hash value of a public key, instead, it is a script source code including operators and operands：scriptPubKey
  - Furthermore, the signature and public key in each input can also be regarded as script source code, including only operands：ScriptSig



2. Transaction in Bitcoin

- **Script of Bitcoin**

Input script (scriptSig)
For current transaction

```
<sig>
<pubKey>
```

Output script (scriptPubKey)
Of some previous TXO serving as the input for current transaction

```
OP_DUP
OP_HASH160
<pubKeyHash?>
OP_EQUALVERIFY
OP_CHECKSIG
```

Output script specify the destination address， input script specify the public key and a signature

To check an input, put the input script of a input and the output script of the referenced TXO together, and execute the operators and operands. If the result is true, it is a valid input.

3. Bitcoin Scripts

- Specified for Bitcoin（inspired by Forth）
- Simple
- Support cryptographic operators/operands
- Based on stack
- No limitation on time and space cost
- No loop instruction（not turning complete）

3. Bitcoin Scripts

operation	functionalities
OP_DUP	Copy the top element of stack
OP_HASH160	Hash two times, the first one uses SHA-256， and the second one usesRIPEMD-160
OP_EQUALVERIFY	Return true if equality holds, retuen false otherwise
OP_CHECKSIG	Verify the signature
OP_CHECKMULTISIG	verify whether a threshold of the signatures is achieved

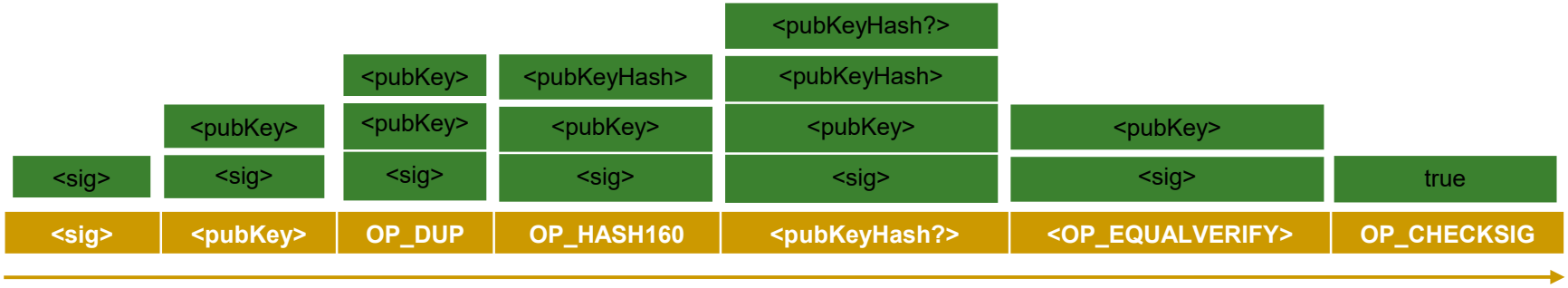
3. Bitcoin Scripts

- example：

<sig>

<pubKey>

OP_DUP
OP_HASH160
<pubKeyHash?>
OP_EQUALVERIFY
OP_CHECKSIG



3. Bitcoin Scripts

- Pay to Script Hash (P2SH)
- The receiver tells the sender ``send your coins to the hash of this script. Impose the condition that to redeem those coins, it is necessary to reveal the script that has the given hash, and further, provide data that will make the script evaluate to true".
- The sender achieves this by using the Pay-to-script-hash (P2SH) transaction type.
  - The P2SH script simply hashes the top value on the stack, checks if it matches the provided hash value, then executes a special second step of validation: that top data value from the stack is reinterpreted as a sequence of instructions, and executed a second time as a script, with the rest of the stack as input.
  - it wasn't part of Bitcoin's initial design specification. It was added after the fact.
- it solves a couple of important problems:
  - It removes complexity from the sender, so the recipient can just specify a hash that the sender sends money to.
  - Miners have to track the set of output scripts that haven't been redeemed yet, and with P2SH outputs, the output scripts are now much smaller as they only specify a hash. All of the complexity is pushed to the input scripts.

3. Bitcoin Scripts: Applications

- ***Escrow transactions***
- ***Efficient micro-payments***
- ***Lock time***

3. Bitcoin Scripts: *Escrow transactions*

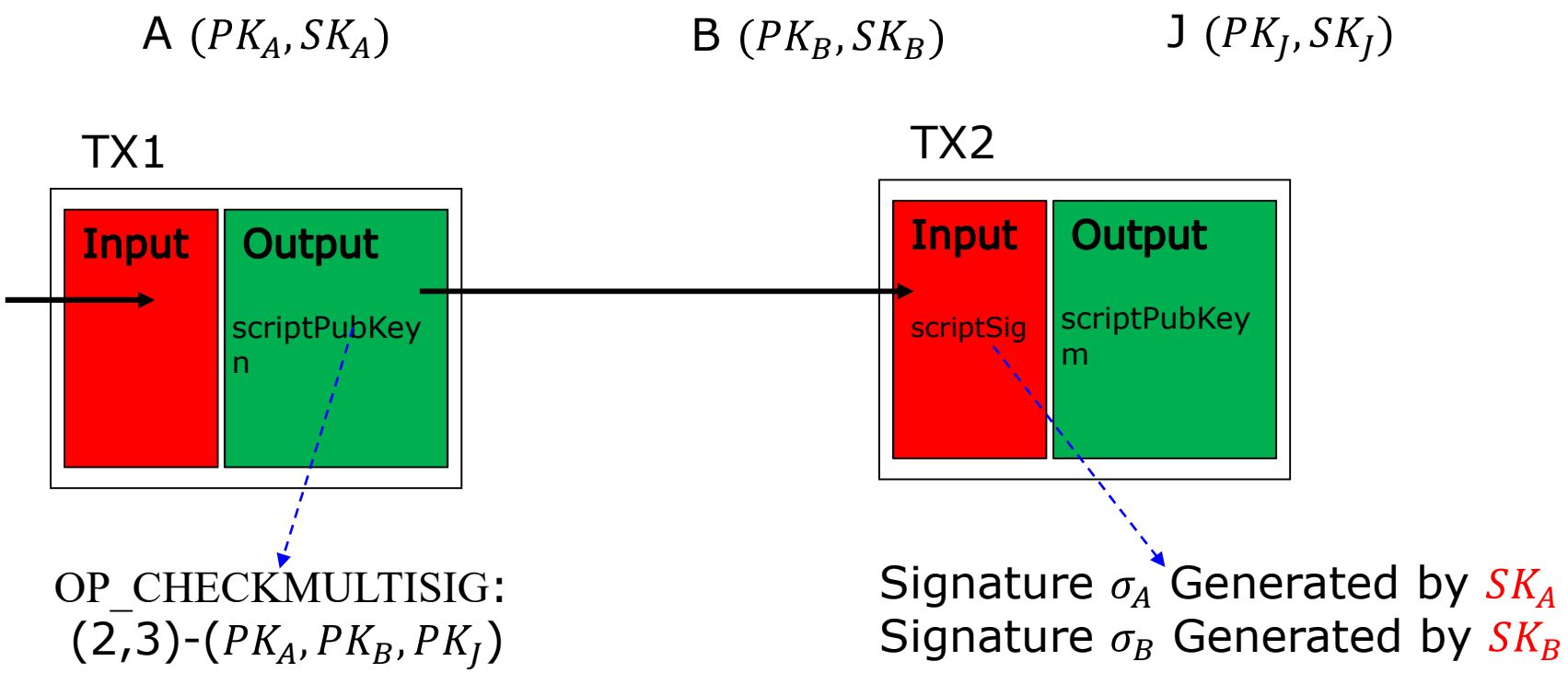
- Alice and Bob want to do business with each other — Alice wants to pay Bob in Bitcoin for Bob to send some physical goods to Alice. The problem though is that Alice doesn't want to pay until after she's received the goods, but Bob doesn't want to send the goods until after he has been paid.
- A nice solution in Bitcoin that's been used in practice is to introduce a third party and do an escrow transaction, using MULTSIG.
  - Alice creates a MULTISIG transaction that requires two of three people to sign in order to redeem the coins. Those three people are going to be Alice, Bob, and some third party arbitrator, Judy, who will come into play in case there's any dispute.
  - This transaction is included in the block chain, and at this point, these coins are held in escrow between Alice, Bob, and Judy, such that any two of them can specify where the coins should go.
  - Bob is convinced that it's safe to send the goods over to Alice, and deliver the goods physically.

3. Bitcoin Scripts: *Escrow transactions*

- Normal Case: Alice and Bob both sign a transaction redeeming the funds from escrow, and sending them to Bob.
- if Bob didn't actually send the goods or they got lost in the mail: Judy needs to get involved. Judy decides between the two possible outcomes.
  - If Judy thinks Bob is cheating, Judy and Alice sign a transaction, sending the money from escrow back to Alice.
  - If Judy thinks Alice is simply refusing to pay when she should, Judy and Bob sign a transaction, sending the money to Bob.
- Judy decides between the two possible outcomes. But the nice thing is that she won't have to be involved unless there's a dispute.

3. Bitcoin Scripts: *Escrow transactions*

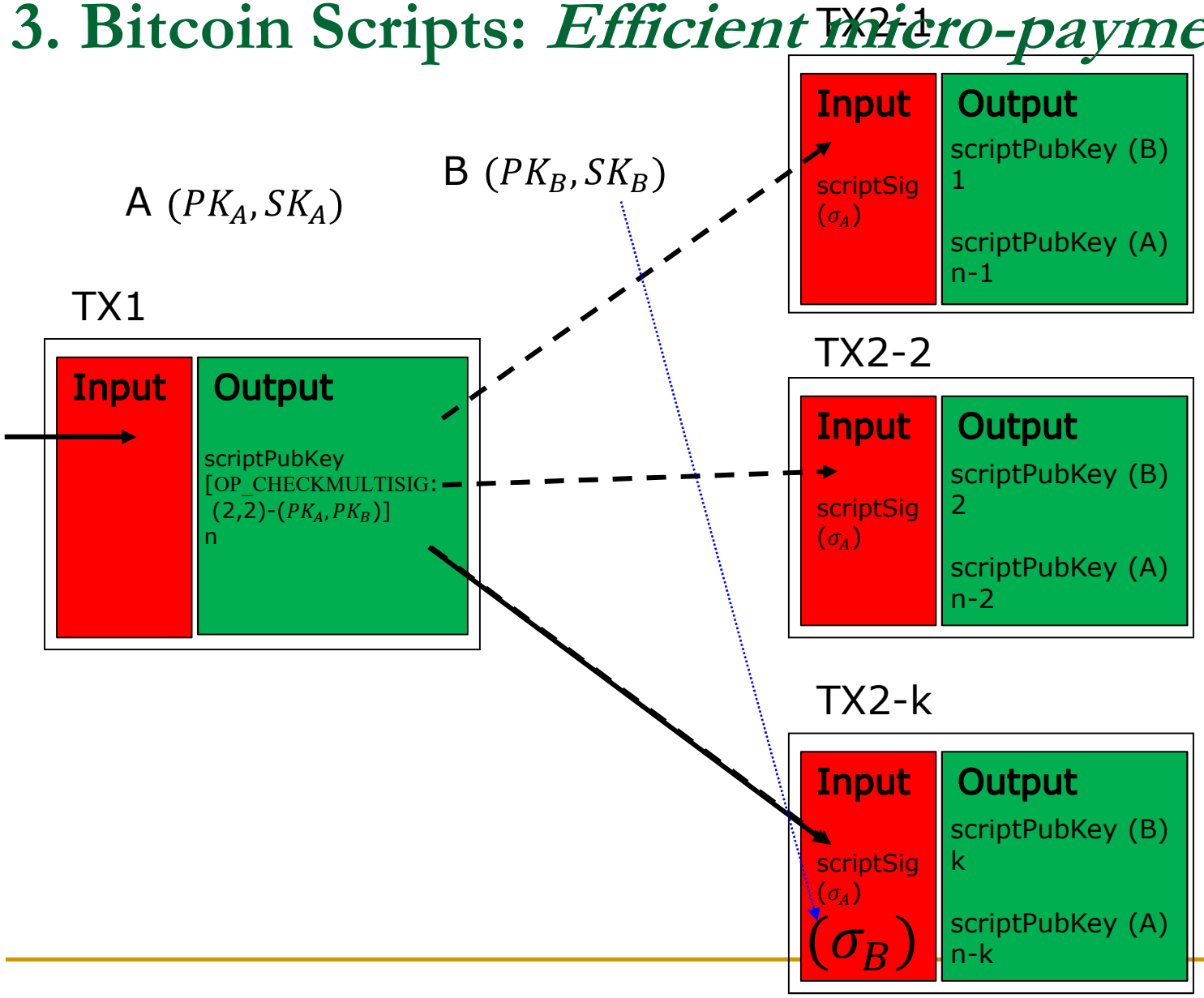
- ***Escrow transactions***



3. Bitcoin Scripts: *Efficient micro-payment*

- Alice is a customer who wants to continually pay Bob small amounts of money for some service that Bob provides. For example, Bob may be Alice's wireless service provider, and requires her to pay a small fee for every minute that she talks on her phone.
  - Creating a Bitcoin transaction for every minute that Alice speaks on the phone won't work. That will create too many transactions, and the transaction fees add up.

3. Bitcoin Scripts: *Efficient micro-payment*



3. Bitcoin Scripts: *Efficient micro-payment*

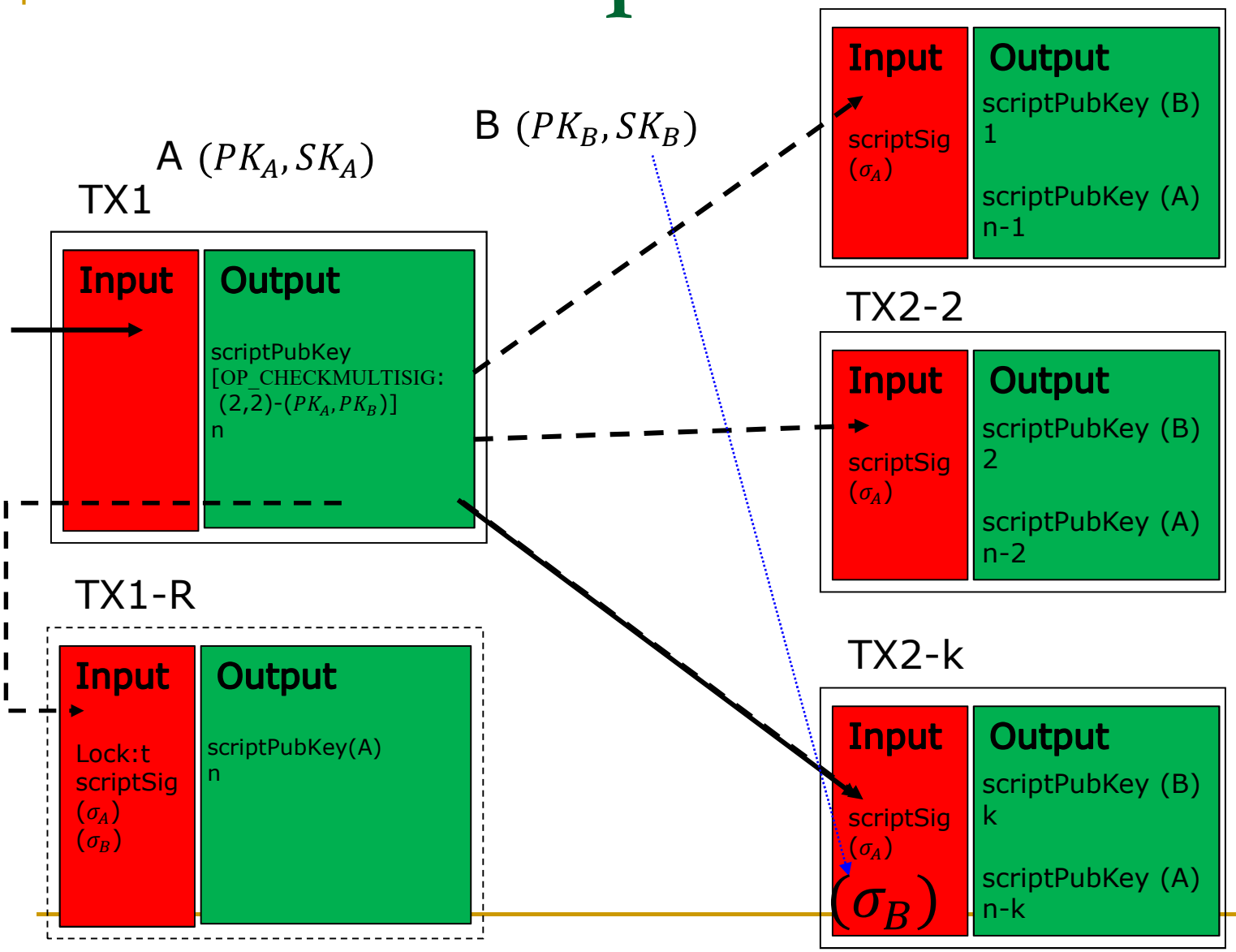
1. Alice creates a MULTISIG transaction paying the maximum anount Alice would ever need to spend to an output, which requires both Alice and Bob to sign to release the coins. The transaction is published to the blockchain.
2. After the first minute that Alice has used the service, she signs a transaction spending those coins that were sent to the MULTISIG address, sending one unit of payment to Bob and returning the rest to Alice.
3. After the next minute of using the service, Alice signs another transaction, this time paying two units to Bob and sending the rest to herself.
4. …… Alice will keep sending these transactions to Bob every minute that she uses the service. (These transactions are not signed by Bob, nor are they being published to the blockchain.)

3. Bitcoin Scripts: *Efficient micro-payment*

- Eventually, Alice will finish using the service, and tells Bob, “I’m done, please cut off my service.” At this point Alice will stop signing additional transactions.
- Bob will disconnect the service, and will take that last transaction that Alice sent him, sign it, and publish that to the block chain.
  - The final transaction that Bob redeems pays him in full for the service that he provided and returns the rest of the money to Alice.
- All those transactions that Alice signed along the way won’t make it to the block chain. Bob doesn’t have to sign them. They’ll just get discarded.

- Problem: what if Bob never signs the last transaction? He may just say, “I’m happy to let the coins sit there in escrow forever,” in which case, maybe the coins won’t move, but Alice will lose the full value that she paid at the beginning.

3. Bitcoin Scripts: *Lock Time*



3. Bitcoin Scripts: *Lock Time*

1. Alice creates a MULTISIG transaction paying the maximum amount Alice would ever need to spend to an output, which requires both Alice and Bob to sign to release the coins. ~~The transaction is published to the blockchain~~.

- Before publishing the MULTISIG transaction, Alice ask Bob to sign a transaction which refunds all of Alice's money back to her, but the refund is "locked" until some time in the future. Alice also sign this refund transaction and hold on to it. Then Alice publish the MULTISIG transaction to the blockchain.

2. After the first minute that Alice has used the service, she signs a transaction spending those coins that were sent to the MULTISIG address, sending one unit of payment to Bob and returning the rest to Alice.
3. After the next minute of using the service, Alice signs another transaction, this time paying two units to Bob and sending the rest to herself.
4. …… Alice will keep sending these transactions to Bob every minute that she uses the service. (These transactions are not signed by Bob, nor are they being published to the blockchain.)

3. Bitcoin Scripts: *Lock Time*

- Eventually, Alice will finish using the service, and tells Bob, “I’m done, please cut off my service.” At this point Alice will stop signing additional transactions. Alice publish the refund transaction to the bitcoin network, with the lock time parameter telling the miners not to publish the transaction until the specified lock time. The transaction will be invalid before either a specific block number, or a specific point in time, based on the timestamps that are put into blocks.
- Bob will disconnect the service, and will take that last transaction that Alice sent him, sign it, and publish that to the block chain.
  - The final transaction that Bob redeems pays him in full for the service that he provided and returns the rest of the money to Alice.
- if Bob does not sign a transaction to redeem pays for his service before the refund transaction becomes valid, all the escrowed amount will be sent back to Alice.
- If Bob honestly sign a transaction to redeem pays to him, the refund transaction will become a double-spending transaction, and miners will discarded it.

Summary

- **1. What is Bitcoin**
- **2. Transaction in Bitcoin**
- **3. Bitcoin Scripts**