

Architecture of solution

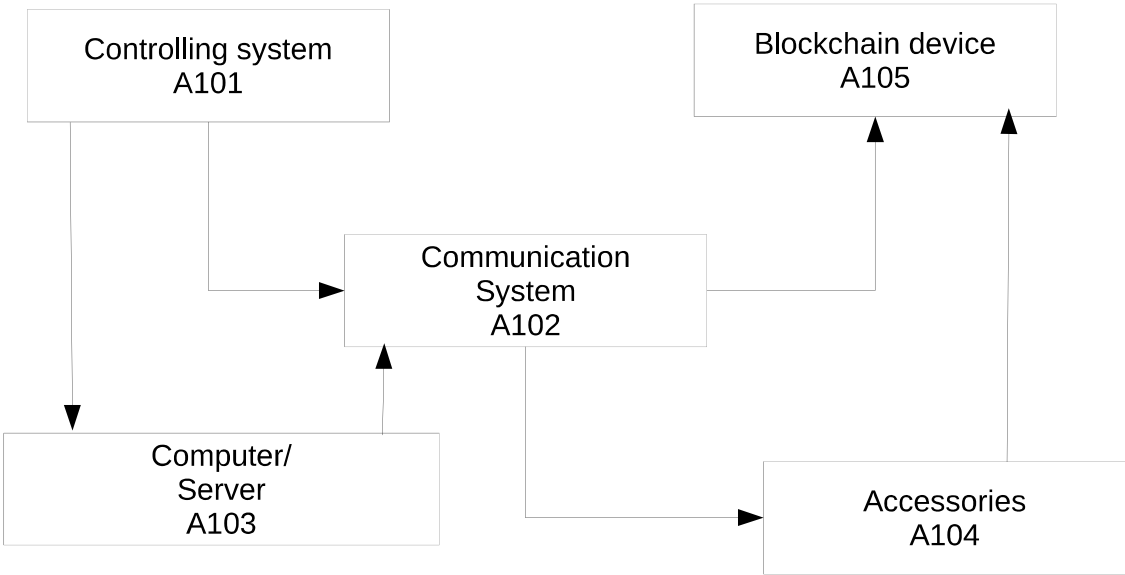


FIG-1A

Example of control codes to be merged at location 10, 20 and end of UID-key.

var Pin = '1234';	A201
var Company = 'ABC';	A202
var Userjson = {	A203
Login:'user1',	
Password : '<password> _{hashhash} '	A204
};	
var Uidkey = keyPair.getPublicKeyBuffer();	A209
var part1 = Uidkey.slice (0,10);	A205
var part2 = Uidkey.slice (10,20);	A206
var part3 = Uidkey.slice (20);	A207
Composite-key plan = part1 + Userjson + part2 +Pin + part3 + Company	A208

FIG-2A

3/23

Example of composite-key creation

var Uidkey = keyPair.getPublicKeyBuffer();	A301
var part1 = Uidkey.slice (0,10);	A302
var part2 = Uidkey.slice (10,20);	A303
var part3 = Uidkey.slice (20);	A304
var Pin = '1234';	A305
var Pinkey = new Buffer(Pin, 'hex');	A306
var Company = 'ABC';	A307
var Companykey = Buffer.from(Company);	A308
var password ='password1';	A309
var passwordbuf = Buffer.from(password);	A310
var passwordhash = bitcoin.crypto.hash160(passwordbuf);	A311
var passwordhashhash = bitcoin.crypto.hash160(passwordhash);	A312
var Userjson = {	A313
Login:'user1',	
Password : passwordhashhash.toString('hex')	A314
};	
var UserPin = JSON.stringify(Userjson);	A315
var Userpinkey = Buffer.from(UserPin);	A316
var compositearray = [new Buffer(part1,'hex'), Userpinkey,	A317
new Buffer(part2, 'hex'), Pinkey,	
new Buffer(part3, 'hex'),	
Companykey];	
var compositekey = Buffer.concat(compositearray);	A318

FIG-3A

Example output of composite-key and redeem-script

Uidkey=02b621acc73f06fbd4cd682a0eda47b6d7317776607a8f02b68a30bf6bdfefbee31	A401
Length=66	A402
part1 0-10= 02b621acc7	A403
part2 10-20= 3f06fbd4cd	A404
lastpart 20-last= 682a0eda47b6d7317776607a8f02b68a30bf6bdfefbee31	A405
compositekey= 02b621acc77b224c6f67696e223a227573657231222c2250617373776f7264223a 22653366333163393138303763336363353832333433613131396538393833343030656661376230 65227d3f06fbd4cd1234682a0eda47b6d7317776607a8f02b68a30bf6bdfefbee31414243	A406
address=2Mui1Rbf6czDe3RozuWKZbMdcKYCMSHQXYG	A407
Pin=1234	A408
Company=ABC	A409
Userjson={"Login":"user1","Password":"e3f31c91807c3cc582343a119e8983400efa7b0e"}	A410
redeemScript=a9144a6ed75b9d255fbefc688482cedbdebfb7a65a3c87	A411

FIG-4A

5/23

Flowchart to create composite-key

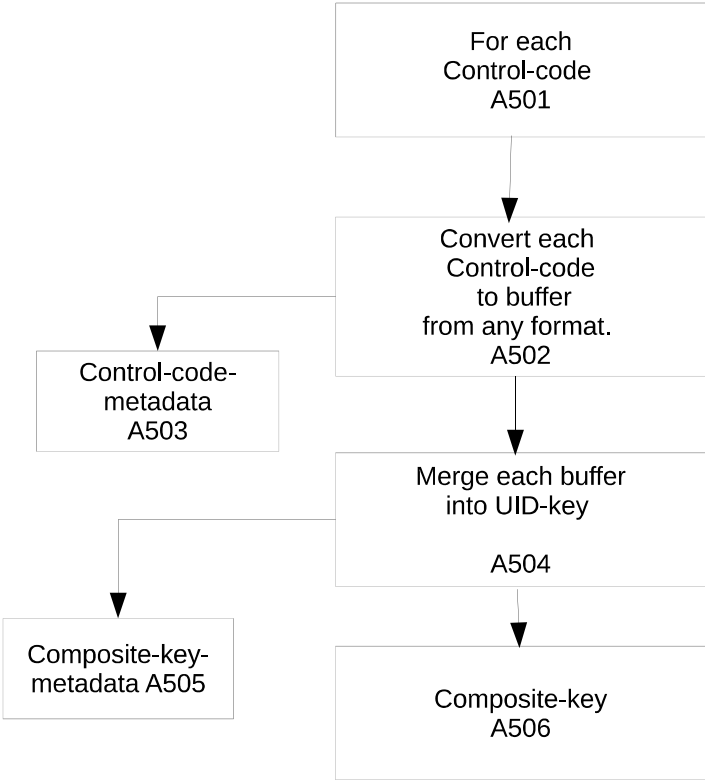


FIG-5A

6/23

Examples of metadata

A602	<pre>Metadata-userjson = { type: 'json' process:'stringify', buffering: 'Buffer', fields : [{name:'Login'}, {name:'Password'}], transformation: [{}, {type_of_hash:'hash160', number_of_times: 2, presented_as: 'hex' }] }</pre>	<pre>Pin=' ' Company=' ' Userjson={"Login": "", "Password": "" }</pre>	A601
A603	<pre>Metadata-company = { type: 'text' process:'none', buffering: 'Buffer', type_of_hash:'none', number_of_times: 0, presented_as: 'text' } Metadata-pin = { type: 'text' process:'none', buffering: 'Buffer', type_of_hash:'none', number_of_times: 0, presented_as: 'text' }</pre>	<pre>Metadata-composite-key = { uidkey_length: 66, parts: 3, partsdesc: [{name: 'part1',start: 0, end:10} {name: 'part2',start:10, end:20} {name: 'part3',start: 20, end:" }], Sequence: ['part1', 'userjson', 'pin', 'part3', 'company'] encryption: 'none', typeof_encryption: 'none' }</pre>	A606
A604			

FIG-6A

7/23

Flowchart to create blockchain address

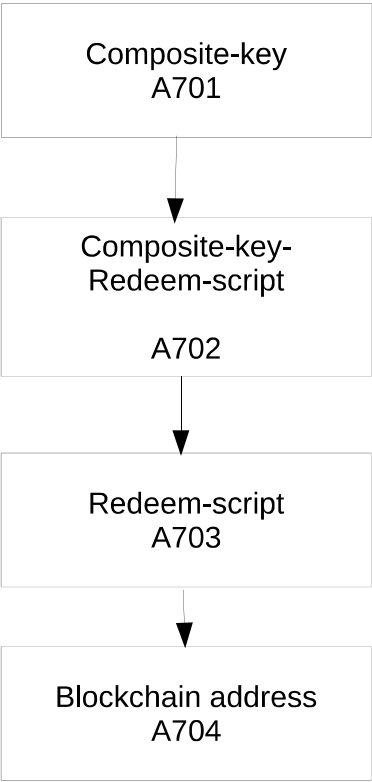


FIG-7A

8/23

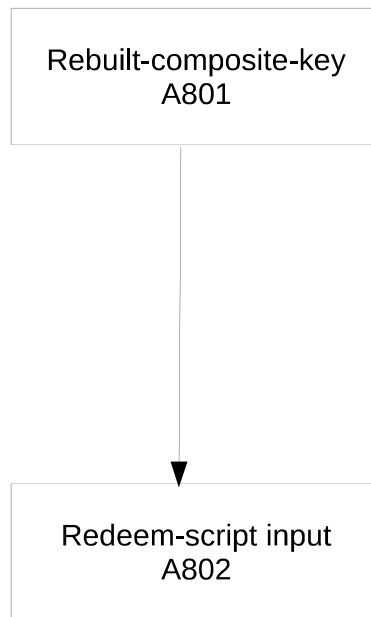
Rebuilt-composite-key used as redeem-script input

FIG-8A

9/23

Communication system

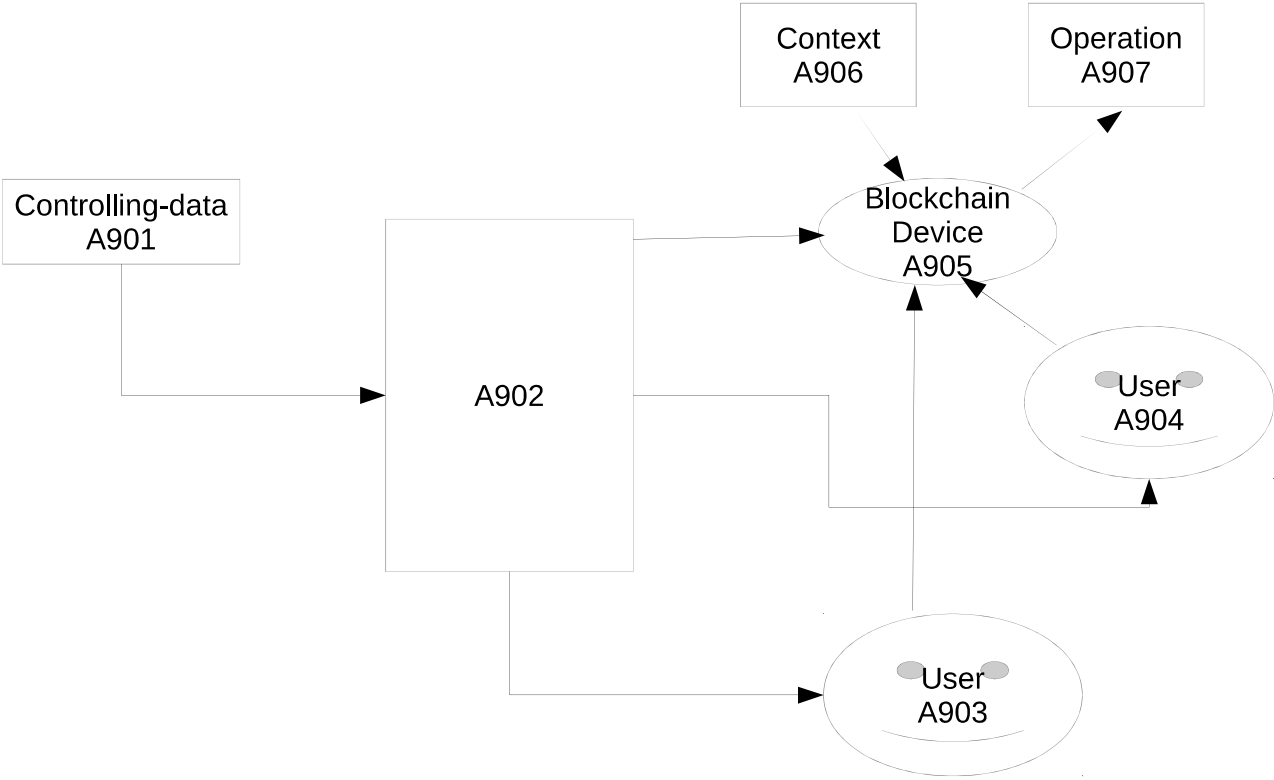


FIG-9A

10/23

Flowchart of Blockchain-device

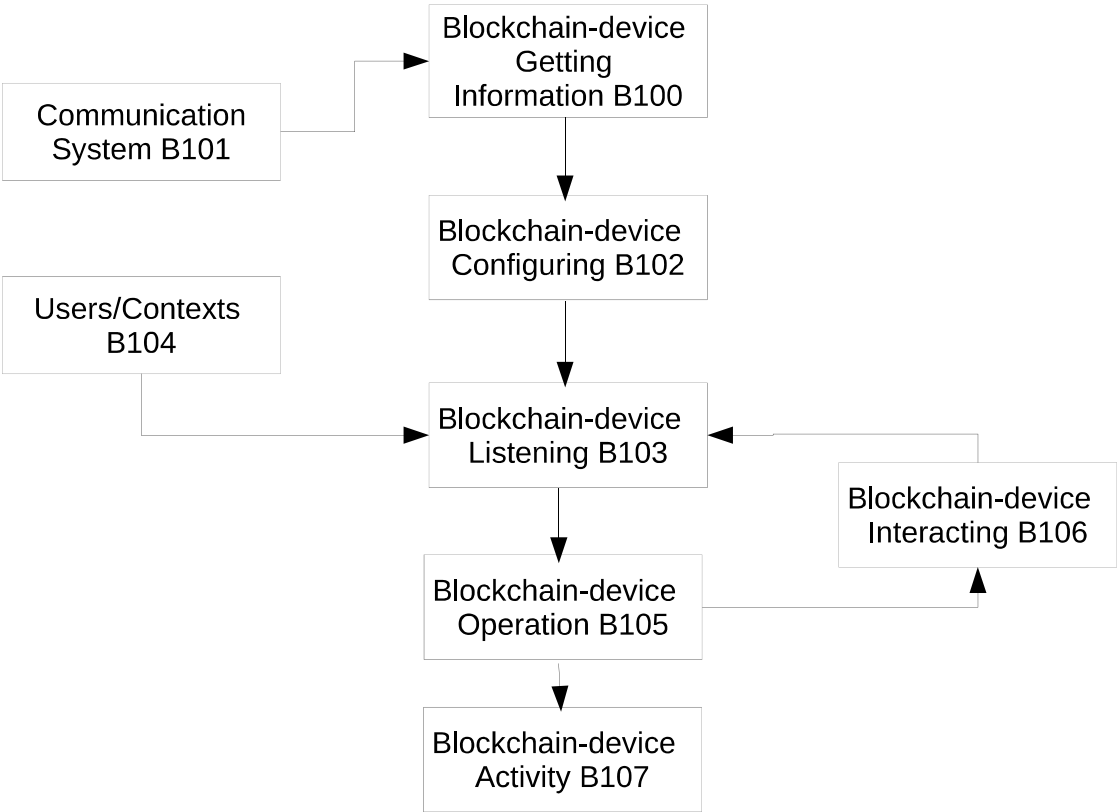


FIG-1B

11/23

Flowchart of blockchain-device operation

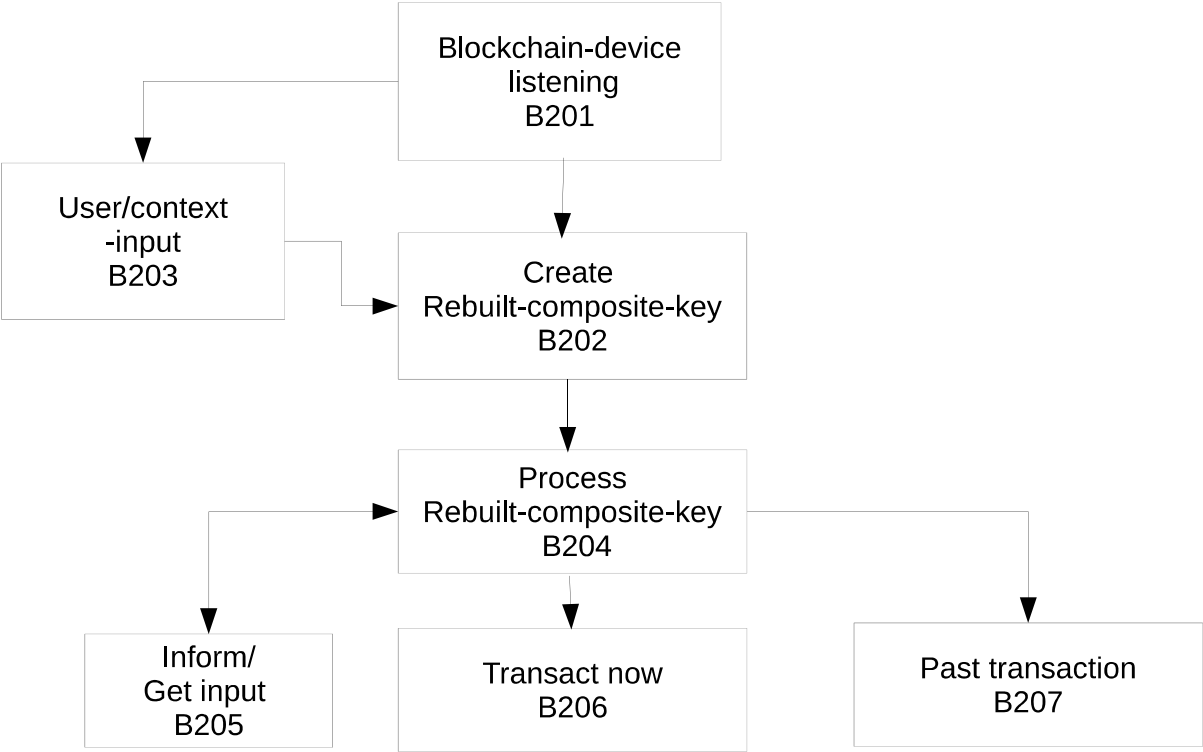


FIG-2B

12/23

Prior-art P2SH and standard redeemScript

scriptPubKey: OP_HASH160 <scriptHash> OP_EQUAL	B301
scriptSig: <input to script> <serialized script>	B302
where <input to script> will be redeem-input	B303
where <serialized script> will be redeemScript	

The standard redeemScript and input to script are

redeemScript: OP_DUP OP_HASH160 <pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG	B304
Where <input to script> is below	B305
<sig> <pubKey>	B306

FIG-3B

P2SH and disclosure redeemScript

scriptPubKey: OP_HASH160 <scriptHash> OP_EQUAL

scriptSig: <input to script> <serialized script>

B401

B402

where <input to script> will be compositeKey

where <serialized script> will be redeemScript

B403

B404

compositeKey will be [<control-codes> <UID-key>]_{processed}

B405

The disclosure redeemScript and input to script are

redeemScript: OP_HASH160 <compositeKeyHash> OP_EQUAL

B406

Where<input to script> is below

<compositeKey>

B407

B408

FIG-4B

14/23

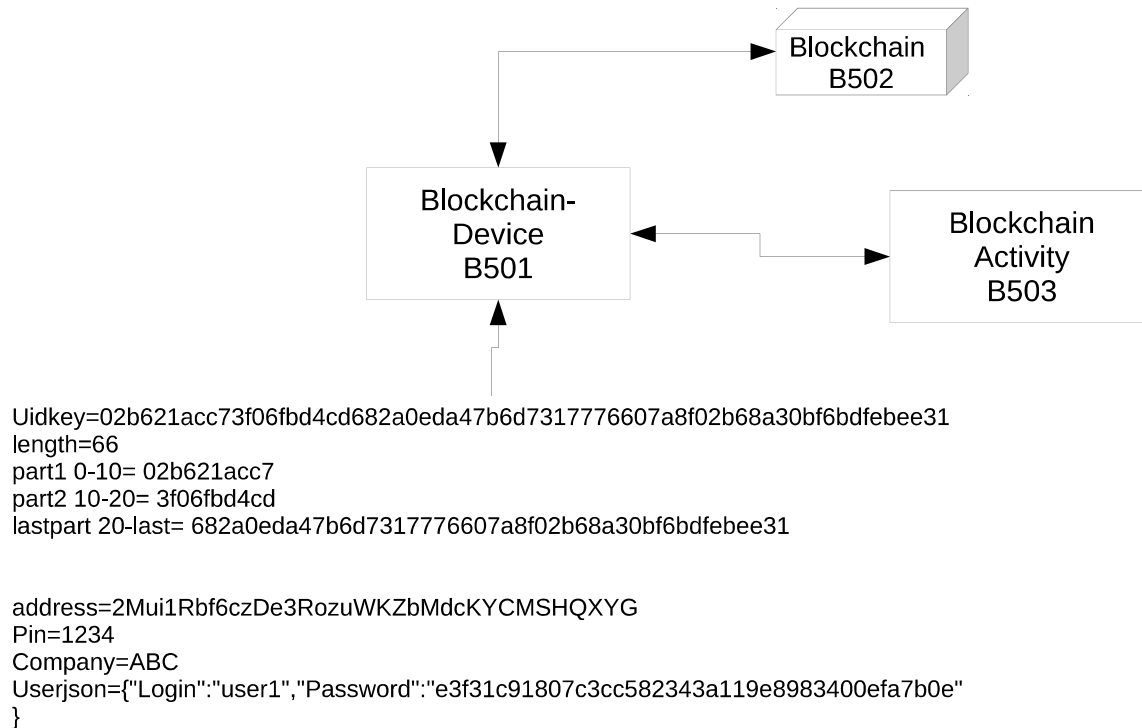
Example of data taken by Blockchain-device for execution

FIG-5B

15/23

Example of data for execution

Uidkey=02b621acc73f06fbd4cd682a0eda47b6d7317776607a8f02b68a30bf6bdfebee31 length=66 part1 0-10= 02b621acc7 part2 10-20= 3f06fbd4cd lastpart 20-last= 682a0eda47b6d7317776607a8f02b68a30bf6bdfebee31	B601
address=2Mui1Rbf6czDe3RoZuWKZbMdcKYCMSHQXYG Pin=1234 Company=ABC Userjson={"Login":"user1","Password":"e3f31c91807c3cc582343a119e8983400efa7b0e"}	B602
rebuilt-compositekey= 02b621acc77b224c6f67696e223a227573657231222c2250617373776f7264223a 22653366333163393138303763336363353832333433613131396538393833343030656661376230 65227d3f06fbd4cd1234682a0eda47b6d7317776607a8f02b68a30bf6bdfebee31414243	B603
redeemScript=a9144a6ed75b9d255fbefc688482cedbdebfb7a65a3c87	B604
https://webbtc.com/script/02b621acc77b224c6f67696e223a227573657231222c2250617373 776f7264223a22653366333163393138303763336363353832333433613131396538393833343030 65666137623065227d3f06fbd4cd1234682a0eda47b6d7317776607a8f02b68a30bf6bdfebee3141 4243%20a9144a6ed75b9d255fbefc688482cedbdebfb7a65a3c87/OP_HASH160%201b00feb96ab25 e224d830a89d10b0341f24e1f4d%20OP_EQUAL	B605

FIG-6B

Example of execution using Rebuilt-composite-key

B701

Bitcoin - Debug Script Execution

Transaction

Not found ([Link to this script](#))

Input Script

02b621acc77b224c6f67696e223a227573657231222c2250617373776f7264223a22653366333163393138303763336363353832333433361313139653839383334303065666137623065227d3f06fbd4cd1234682a0eda47b6d7317776607a8f02b68a30bf6bdfabee31414243

Output Script

OP_HASH160 1b00feb96ab25e224d830a89d10b0341f24e1f4d OP_EQUAL

Signature Hash

Optional: If empty, all signatures are assumed to be valid.

Verify

☒ minimaldata ☒ sigpushonly ☒ cleanstack ☒ dersig ☒ low_s ☒ strictenc (hover for info)

Result

true

(Re-)Run Script

Inner P2SH Script

OP_HASH160
4a6ed75b9d255fbefc688482cedbdebfb7a65a3c
OP_EQUAL

Addresses / Data
[38Uai88M6blqvRZ6pWV05w4TZqEw5VCL0n](#)

FIG-7B

System architecture of solution

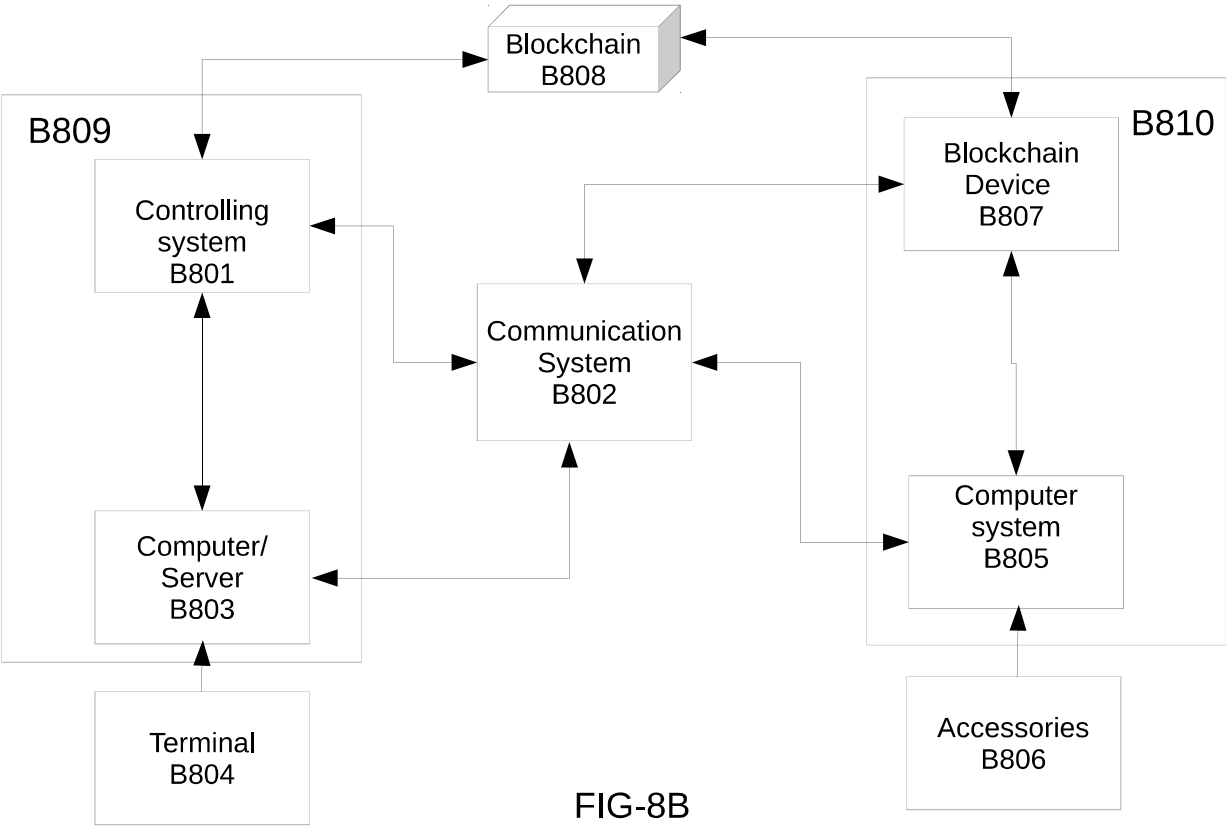


FIG-8B

Service architecture of solution

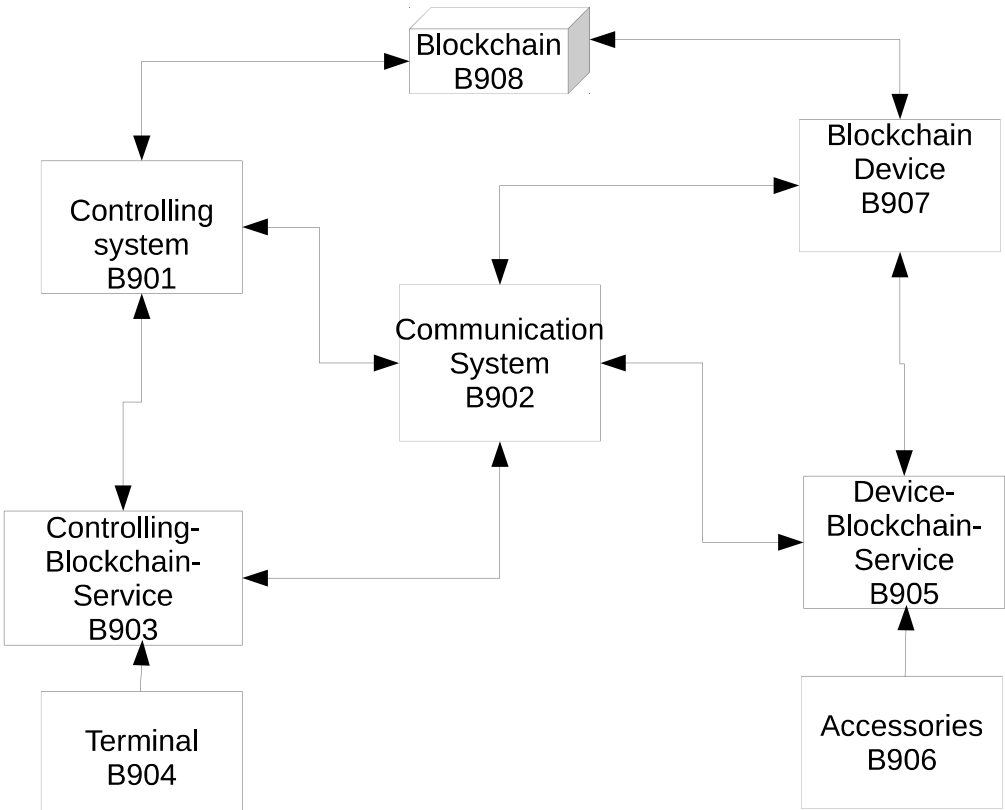


FIG-9B

Simple usecase of solution

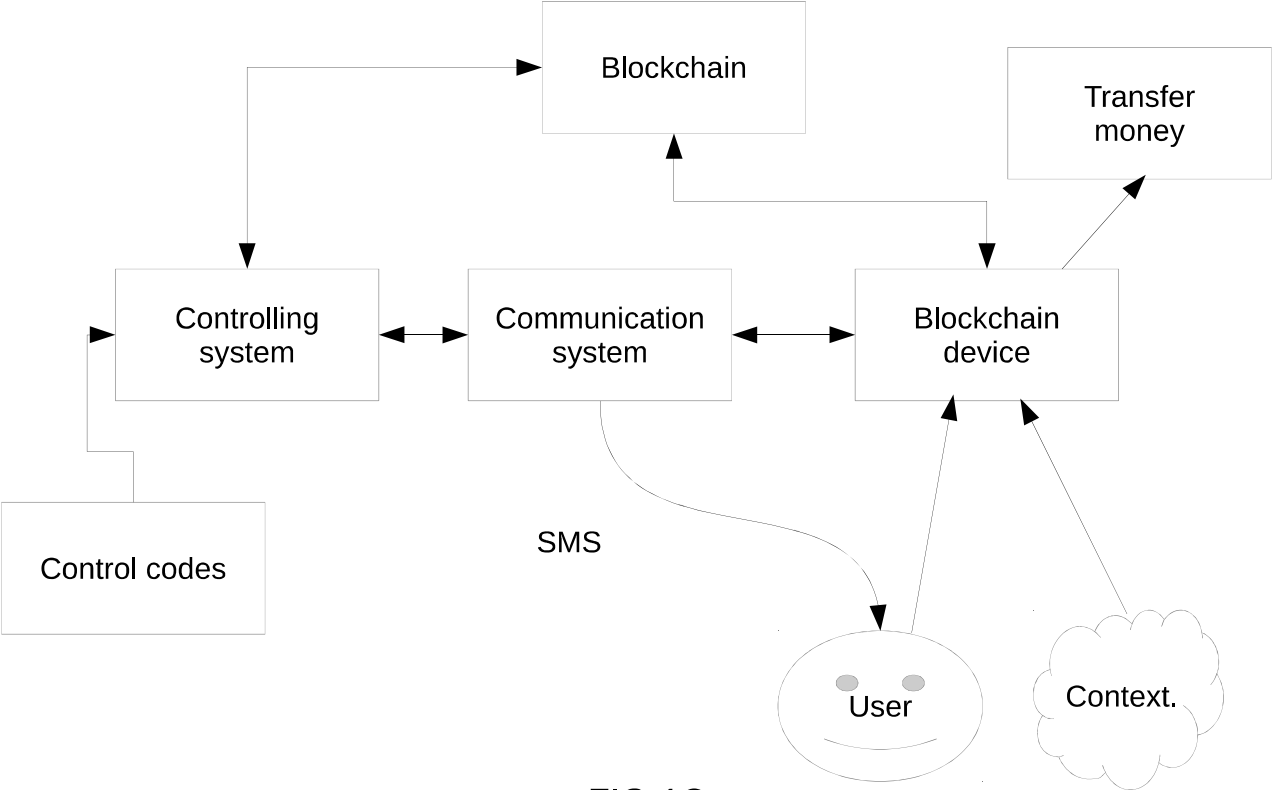


FIG-1C

20/23

Steps for using custom-contract

CompositeRedeemScript = [bitcoin.opcodes.OP_HASH160, compositekeyHash, bitcoin.opcodes.OP_EQUAL]	C201
VerifyCompositeOutput = [bitcoin.opcodes.OP_VERIFY] ;	C202
RedeemScript = <CompositeRedeemScript>.concat(<VerifyCompositeOutput>,<custom-contract>)	C203

FIG-2C

21/23

Javascript example of using custom-contract

```
bitcoin = require('bitcoinjs-lib');
var keyPair = bitcoin.ECPair.makeRandom(bitcoin.networks.testnet);

var Uidkey = keyPair.getPublicKeyBuffer();
var OTP = '6784';
var Pinkey = new Buffer(OTP, 'hex');
var arr = [Pinkey, Uidkey];
var compositekey = Buffer.concat(arr);

var compositeKeyHash = bitcoin.crypto.hash160(compositekey);

var Pin1 = '1234';
var Pin1Hash = bitcoin.crypto.hash160(new Buffer(Pin1, 'hex'));

var Pin2 = 'mypassword12';
var Pin2buffer = bitcoin.crypto.hash160(Buffer.from(Pin2));

var Pin2Hash = bitcoin.crypto.hash160(Pin2buffer);

var customcontract =
[bitcoin.opcodes.OP_HASH160, Pin1Hash, bitcoin.opcodes.OP_EQUALVERIFY,
 bitcoin.opcodes.OP_HASH160, Pin2Hash, bitcoin.opcodes.OP_EQUAL];

var redeemScript =
    bitcoin.script.compile([bitcoin.opcodes.OP_HASH160, compositeKeyHash,
    bitcoin.opcodes.OP_EQUAL].concat([bitcoin.opcodes.OP_VERIFY],
    customcontract));

var scriptPubKey = bitcoin.script.scriptHashOutput(bitcoin.crypto.hash160(redeemScript))

var address = bitcoin.address.fromOutputScript(scriptPubKey, bitcoin.networks.testnet)
```

FIG-3C

Example of transaction with custom-contract

address=2N7DZ1iasLDFrJ2s4PujTTphUx11qKyhrq7	C401
(control-code) OTP=6784	C402
(data for contract-input)	
Pin1=1234	C403
Pin2=mypassword12	C404
redeemScript=a9144900b7db35741f5cb7f6eb1ea3c708cf7ac4e09e88a914c9440a6ff7e66be6ce4a3524a7c44f292a63153388a914aa5e8c44612ea01b64f65b597bfbe3f670ff7ed487	C405
Uidkey=025dbd1b04f5c708e9f7775649ff89ec03dfef34bfad94b8f41c0e450f37b9fff5	C406
compositeKey=6784025dbd1b04f5c708e9f7775649ff89ec03dfef34bfad94b8f41c0e450f37b9fff5	C407
compositeKeyHash=4900b7db35741f5cb7f6eb1ea3c708cf7ac4e09e	C408
combinedInput=6dd3b9a74b883a87596e3dfe495bc23b33425214%201234%206784025dbd1b04f5c708e9f7775649ff89ec03dfef34bfad94b8f41c0e450f37b9fff5	C409
https://webbtc.com/script/6dd3b9a74b883a87596e3dfe495bc23b33425214%201234%206784025dbd1b04f5c708e9f7775649ff89ec03dfef34bfad94b8f41c0e450f37b9fff5%20a9144900b7db35741f5cb7f6eb1ea3c708cf7ac4e09e88a914c9440a6ff7e66be6ce4a3524a7c44f292a63153388a914aa5e8c44612ea01b64f65b597bfbe3f670ff7ed487/OP_HASH160%209940cf61d906e99020d685cb6b836953a12c5e29%20OP_EQUAL	C410

FIG-4C

Example of execution using custom-contract

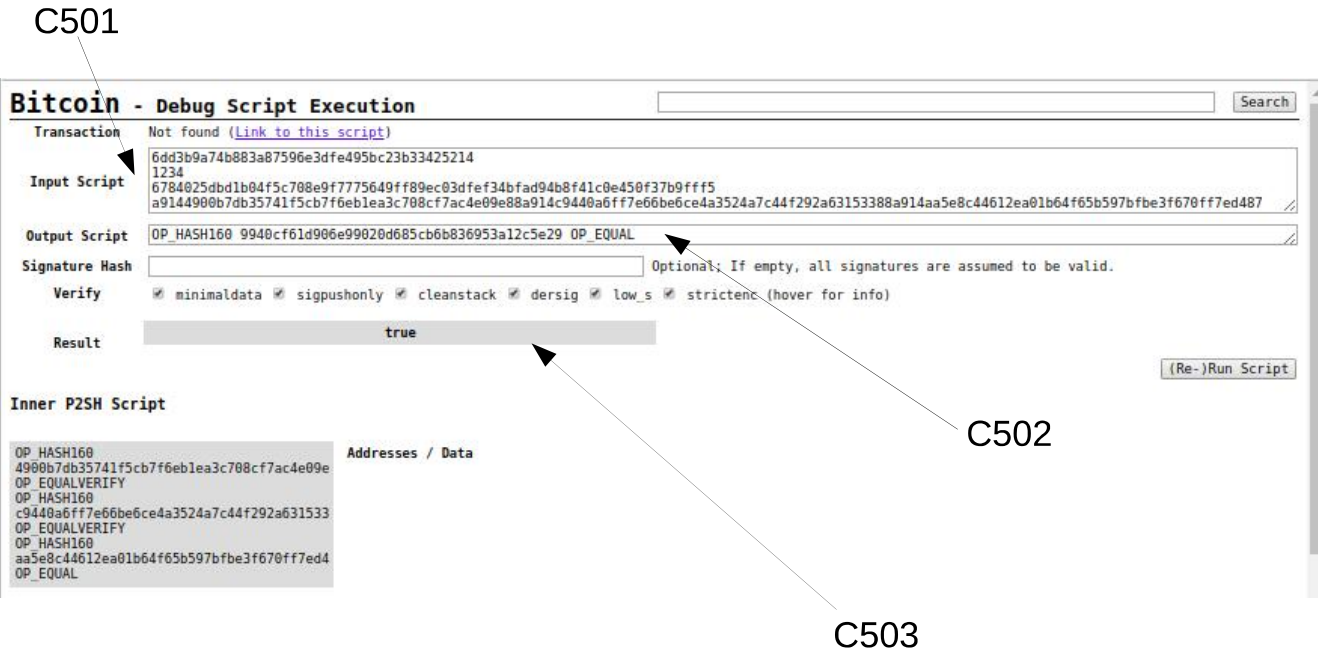


FIG-5C