| 步骤 | 实施位置 | 输入用例 | 输出用例 | 详细说明 |
| --- | --- | --- | --- | --- |
| account create | client | N/A | keypair  {x: 9494a5097b4ac71c432e485d08f89336327ce8091d6bfb3f0e2c46c7e115cf8, y:[‘0x0c1a7b885840975ed841fed9353aee7ae55645008b39dceb22d868828443dbbf’, ‘0x0d7c0a4c5ce3692ba2d1f19b6b74e5b9ad2068511443bd98a4210dfd0843c246’]} | Generate keypair with ECC  const x = bn128.randomScalar();  const y = bn128.curve.g.mul(x);  return { 'x': x, 'y': y }; |
| account register input compute | client | (zscAddress, keypair)  zscAddress: zether smart contract Address  ‘0xa82811850684Ea34AA2Ce0aDfD1A3A864a6b7e3c’ | [c,s]  [  '0x300bc8955b61d44b6e25da9c1af8ffa726edced74825b3667a8344bfb1f3326f',  '0x2bd12f1cbbe77041ba6ca8f97f6c59baf16384626ec4603d6d74fe7bc92209bc'  ] | sign the zscAddress with the keypair  const k = bn128.randomScalar();  const K = bn128.curve.g.mul(k);  const c = utils.hash(ABICoder.encodeParameters([  'address',  'bytes32[2]',  'bytes32[2]',  ], [  address,  bn128.serialize(keypair['y']),  bn128.serialize(K),  ]));  const s = c.redMul(keypair['x']).redAdd(k);  return [bn128.bytes(c), bn128.bytes(s)]; |
| account register | smart contract | function register(Utils.G1Point memory y, uint256 c, uint256 s)  y: keypair y  [‘0x0c1a7b885840975ed841fed9353aee7ae55645008b39dceb22d868828443dbbf’, ‘0x0d7c0a4c5ce3692ba2d1f19b6b74e5b9ad2068511443bd98a4210dfd0843c246’]  [c,s] sign:  [  '0x300bc8955b61d44b6e25da9c1af8ffa726edced74825b3667a8344bfb1f3326f',  '0x2bd12f1cbbe77041ba6ca8f97f6c59baf16384626ec4603d6d74fe7bc92209bc'  ] | N/A  update the pending  pending[yHash][0] = y;  pending[yHash][1] = Utils.g();  [  '0x2152b3c0aa1d7d04722fdde1bcde833c381db04002abc6e8385157fad062a707',  '0x2bc8797d4b5e151ce6800d142cf02f46603b48440d555b8ec863a4589604d68b',  x: '0x2152b3c0aa1d7d04722fdde1bcde833c381db04002abc6e8385157fad062a707',  y: '0x2bc8797d4b5e151ce6800d142cf02f46603b48440d555b8ec863a4589604d68b'  ] | verify the signature with address, y, c, s  Utils.G1Point memory K = Utils.g().mul(s).add(y.mul(c.neg()));  uint256 challenge = uint256(keccak256(abi.encode(address(this), y, K))).mod();  require(challenge == c, "Invalid registration signature!"); |
| account deposit | smart contract | function fund(Utils.G1Point memory y, uint256 bTransfer)  y: keypair y  [‘0x0c1a7b885840975ed841fed9353aee7ae55645008b39dceb22d868828443dbbf’, ‘0x0d7c0a4c5ce3692ba2d1f19b6b74e5b9ad2068511443bd98a4210dfd0843c246’]  bTransfer: deposit amount | N/A  update the pending  Utils.G1Point memory scratch = pending[yHash][0];  scratch = scratch.add(Utils.g().mul(bTransfer));  pending[yHash][0] = scratch;  [  '0x28768e6642892f96430210a80835c7ba27d2d56c89cd3a78d304e6e7659697e5',  '0x16252d548b01ab32ab9a480827d6f21321f592fbcb458b7ac3bfeef856431da6',  x: '0x28768e6642892f96430210a80835c7ba27d2d56c89cd3a78d304e6e7659697e5',  y: '0x16252d548b01ab32ab9a480827d6f21321f592fbcb458b7ac3bfeef856431da6'  ] | deposit some fund to the zether smart contract.  coin.transferFrom(msg.sender, address(this), bTransfer), |
| account withdraw input compute | client | y: keypair y  bTransfer: withdraw amount  lastRollOver: Epoch  home: withdraw eth address  '0xEdBE569d156997aa6B292035B85322dfaEf0A59a' | C: ElGraml value for withdraw  const C  = deserialized.plus(new BN(-value));  proof: Zk Proof for burn  const proof = Service.proveBurn(  C,  account.keypair["y"],  state.lastRollOver,  home,  account.keypair["x"],  state.available - value  )  BA: diff  BS: random  T\_1  T\_2  tHat  mu  c  s\_sk  s\_b  s\_tau  ipProof    u: epoch sign  const u = utils.u(state.lastRollOver, account.keypair["x"]);  u: [  '0x28c85c43a3be2d9859d8285dcee66191c45b5014c094fd308829ec1a7bdbbdfa',  '0x22c23f9993b0d29ee4c52cf08c7275ffa8e767f9f0aa908c9a6d14fdd034d6d8',  x: '0x28c85c43a3be2d9859d8285dcee66191c45b5014c094fd308829ec1a7bdbbdfa',  y: '0x22c23f9993b0d29ee4c52cf08c7275ffa8e767f9f0aa908c9a6d14fdd034d6d8'  ] |  |
| account withdraw | smart contract | function burn(  Utils.G1Point memory y,  uint256 bTransfer,  Utils.G1Point memory u,  bytes memory proof  )  y: [  '0x2152b3c0aa1d7d04722fdde1bcde833c381db04002abc6e8385157fad062a707',  '0x2bc8797d4b5e151ce6800d142cf02f46603b48440d555b8ec863a4589604d68b',  x: '0x2152b3c0aa1d7d04722fdde1bcde833c381db04002abc6e8385157fad062a707',  y: '0x2bc8797d4b5e151ce6800d142cf02f46603b48440d555b8ec863a4589604d68b'  ],  bTransfer: ‘10’  u: epoch sign  u: [  '0x28c85c43a3be2d9859d8285dcee66191c45b5014c094fd308829ec1a7bdbbdfa',  '0x22c23f9993b0d29ee4c52cf08c7275ffa8e767f9f0aa908c9a6d14fdd034d6d8',  x: '0x28c85c43a3be2d9859d8285dcee66191c45b5014c094fd308829ec1a7bdbbdfa',  y: '0x22c23f9993b0d29ee4c52cf08c7275ffa8e767f9f0aa908c9a6d14fdd034d6d8'  ]  proof: zk proof for burn   | N/A  update pending  Utils.G1Point[2] memory scratch = pending[yHash];  pending[yHash][0] = scratch[0].add(Utils.g().mul(bTransfer.neg()));  [  '0x1b7721f698852abd50a3761fd5cae960df1f78d65f8cf9655a71ab6c0911b67c',  '0x0d24232b82330d9dbdffe458c3ade576f55af32aab92b67dd5a3b8a1d9407543',  x: '0x1b7721f698852abd50a3761fd5cae960df1f78d65f8cf9655a71ab6c0911b67c',  y: '0x0d24232b82330d9dbdffe458c3ade576f55af32aab92b67dd5a3b8a1d9407543'  ]  verify proof  burnVerifier.verifyBurn(  scratch[0],  scratch[1],  y,  lastGlobalUpdate,  u,  msg.sender,  proof  ), | withdraw the amount from the contract to user  coin.transfer(msg.sender, bTransfer), |
| account transfer input compute | client | name: transfer user  “alice”  value: amount  10  decoys: annoy group  “bod”,”dave”  beneficiary: miner  “Miner”  const proof = Service.proveTransfer(  Cn,  C,  y,  state.lastRollOver,  account.keypair["x"],  r,  value,  state.available - value - fee,  index,  fee  )  Cn: smart contract acc value with C change  C: change value  y: account  lastRollOver epoch  x: private key  r: random key  value: amount change value  state.available - value -fee: remain value  index: alice account index  fee: fee amount | C:  all transfer value in Elgamel value with random D  D:  const r = bn128.randomScalar();  const D = bn128.curve.g.mul(r);  random point  y[]:  public address  u:  rollOver sign with x  proof:  zk proof  beneficiary:  miner address |  |
| account transfer | smart contract | function transfer(  Utils.G1Point[] memory C,  Utils.G1Point memory D,  Utils.G1Point[] memory y,  Utils.G1Point memory u,  bytes memory proof,  Utils.G1Point memory beneficiary  )  C:  [  [  '0x16cf3bc6cd7c63a9f0cfd3d9351438b4c14fb9fa338840ea56edee0912ee1b72',  '0x26c8715f92eef2ac67529e8159978165e269171736a46a23b6175ac6ccc3b48e',  x: '0x16cf3bc6cd7c63a9f0cfd3d9351438b4c14fb9fa338840ea56edee0912ee1b72',  y: '0x26c8715f92eef2ac67529e8159978165e269171736a46a23b6175ac6ccc3b48e'  ],  [  '0x2ac1aab404c50f8e30109794962dc187bbdf436de60eec75655bbff089b7e1d6',  '0x06461e89f8256b9d9b2f185f5c1a8228a7db7522e13bf99b301122d9b924fe25',  x: '0x2ac1aab404c50f8e30109794962dc187bbdf436de60eec75655bbff089b7e1d6',  y: '0x06461e89f8256b9d9b2f185f5c1a8228a7db7522e13bf99b301122d9b924fe25'  ],  [  '0x1fc50df91ce642424b1eedd793cb7dd4f0093931aacdcd2d4266a4dd390edebd',  '0x191364ed1db014023d2fc9480d57b84721908b36fafa94b8bf19b8d21b6d2f67',  x: '0x1fc50df91ce642424b1eedd793cb7dd4f0093931aacdcd2d4266a4dd390edebd',  y: '0x191364ed1db014023d2fc9480d57b84721908b36fafa94b8bf19b8d21b6d2f67'  ],  [  '0x0d0d9dca2c942dce63234f40431d70e2cdeed94c0db15bfe70432621917aaa49',  '0x256e0106734c1a88a996c3f3161f9de3974288d62582cfafc11408fbf0e91545',  x: '0x0d0d9dca2c942dce63234f40431d70e2cdeed94c0db15bfe70432621917aaa49',  y: '0x256e0106734c1a88a996c3f3161f9de3974288d62582cfafc11408fbf0e91545'  ]  ]  D:  [  '0x007ced2c73ba3afb14a79da9c9d66f9c0a19256e8f3f16d650011ddca3cbb798',  '0x15dd7569df8333e67a7ef2874f4c2031f705f116bdaf19be3c72ff7de7fd8e9a',  x: '0x007ced2c73ba3afb14a79da9c9d66f9c0a19256e8f3f16d650011ddca3cbb798',  y: '0x15dd7569df8333e67a7ef2874f4c2031f705f116bdaf19be3c72ff7de7fd8e9a'  ]  y: public key  [  [  '0x05d44242dcc9a9fedae1718ceaadc35eab9559f70fb7715f74bc01a487b49452',  '0x2e88e64131cda8b63af3fee39691c5129116acd1b6642a32e37a930e2b30d67c',  x: '0x05d44242dcc9a9fedae1718ceaadc35eab9559f70fb7715f74bc01a487b49452',  y: '0x2e88e64131cda8b63af3fee39691c5129116acd1b6642a32e37a930e2b30d67c'  ],  [  '0x19e0b126e3ca80ba889467492f558aecde5ba3a5f151fefac612861e733ee19f',  '0x1d0ed3103a8e0f6396f00a859d98fef863418aac20da137033700c741f08226d',  x: '0x19e0b126e3ca80ba889467492f558aecde5ba3a5f151fefac612861e733ee19f',  y: '0x1d0ed3103a8e0f6396f00a859d98fef863418aac20da137033700c741f08226d'  ],  [  '0x19c0270ce6ef2ee2b65103f6cfa90a1dd503011c24d88f541b6ee9e8ca0d454e',  '0x22bba3f93e9998ddc95f7811478143856c781ff146fad8edda50493ead146960',  x: '0x19c0270ce6ef2ee2b65103f6cfa90a1dd503011c24d88f541b6ee9e8ca0d454e',  y: '0x22bba3f93e9998ddc95f7811478143856c781ff146fad8edda50493ead146960'  ],  [  '0x2152b3c0aa1d7d04722fdde1bcde833c381db04002abc6e8385157fad062a707',  '0x2bc8797d4b5e151ce6800d142cf02f46603b48440d555b8ec863a4589604d68b',  x: '0x2152b3c0aa1d7d04722fdde1bcde833c381db04002abc6e8385157fad062a707',  y: '0x2bc8797d4b5e151ce6800d142cf02f46603b48440d555b8ec863a4589604d68b'  ]  ]  u:  [  '0x15ce7f9fad945a2db3a5fd46fa3a261e0f4f5eae0641dcb57560a6d9850114d5',  '0x2c7c259884b62f262a6abc33fc443cecb51c5d581df2027a73538d626c201feb',  x: '0x15ce7f9fad945a2db3a5fd46fa3a261e0f4f5eae0641dcb57560a6d9850114d5',  y: '0x2c7c259884b62f262a6abc33fc443cecb51c5d581df2027a73538d626c201feb'  ]  proof:    beneficiary:  [  '0x0607f45ce8a91e43fb804f92e22108d7e14e40556f040fb509de3271f5400689',  '0x2e858b20cdd5cdaffcbbd4275b9c44fa2f5b43f85e9b720668c92470b8345f46',  x: '0x0607f45ce8a91e43fb804f92e22108d7e14e40556f040fb509de3271f5400689',  y: '0x2e858b20cdd5cdaffcbbd4275b9c44fa2f5b43f85e9b720668c92470b8345f46'  ] | N/A  update pending  Utils.G1Point[2] memory scratch = pending[yHash];  pending[yHash][0] = scratch[0].add(C[i]);  pending[yHash][1] = scratch[1].add(D);  verify proof  zetherVerifier.verifyTransfer(  CLn,  CRn,  C,  D,  y,  lastGlobalUpdate,  u,  proof  ) | transfer the amount from alice to bob in zether |