advanced MPC

Ge Zhonghui

1 data structure

1.1 payment

pay = (type, round, aux)

- type = dirp, aux = (pr, pe, amt) — pr-payer(付款方), pe-payee(收款方), amt-支付金额
- type = conp, aux = (pr, pe, amt, h, time)
- type = fulp, aux = (cmd, cpay) — cmd = complete / cancel, cpay-fulp 针对的条件支付

为方便描述，pe:=pay.other(pr), pr:=pay.other(pe)

1.2 input

$Input_j = (pay_j, bal_j, ConPay_j)$ 为方便描述，$pay_j.round$ 也记为 $Input_j.round.$
签名验证：$Vrf_j(m, \sigma)$ 表示用 P_j 的公钥对消息 m 及签名 σ 进行验证。验证通过返回 1，失败返回 0.

1.3 state

$(state_r = (r, \{Input_j, \sigma_j\}_j, F)), \sigma_{cus})$
其中，σ_j 为 P_j 对 $Input_j$ 的签名，σ_{cus} 为 custodian 对 $state_r$ 的签名。
签名验证：$Vrf_{cus}(state_r, \sigma_{cus})$ 表示用 custodian 的公钥对 $state_r$ 及签名 σ_{cus} 进行验证。验证通过返回 1，失败返回 0.

2 Audit

2.1 Audit single state

对于单个 $state_r$，定义以下事件，

- ε_1. 存在 $\{Input_j, \sigma_j\} \in state$, 且 $Vrf_j(Input_j, \sigma_j) = 0$
- ε_2. 存在 $bal_j < 0$, 或者 $\sum_j(bal_j + 1/2cpay.amt, cpay \in ConPay_j) \neq totalValue$
- ε_3. 存在 $cpay \in ConPay_j$, set $P_i := cpay.other(P_j)$, $cpay \notin ConPay_i$
- ε_4. 存在 pay_j, set $P_i := pay_j.other(P_j)$, 使得 $(pay_j.round > r) \vee (pay_i.round > r) \vee (pay_j.round > pay_i.round) \vee ((pay_j.round = pay_i.round) \wedge (pay_j \neq pay_i))$

Algorithm 1: AuditSin

Input: $state$

Output: Correctness of $state$, True or False

```
if (ε1 ∨ ε2 ∨ ε3 ∨ ε4) then
    return False
else
    return True
```

2.2 Audit double states

2.3 Audit()

写在合约中的 Audit() 还需考虑 custodian 对 state 的签名，和 bestState 这种没有签名的特殊情况。

Algorithm 2: AuditDou

Input: $(state_{r_1}, state_{r_2})$, assume $r_1 \geq r_2$

Output: Correctness of states, True or False

parse $state_r$ as $(r, \{Input_{r,j}, \sigma_{r,j}\}_j, F_r), r \in \{r_1, r_2\}$
parse $Input_{r,j}$ as $(pay_{r,j}, bal_{r,j}, ConPay_{r,j}), r \in \{r_1, r_2\}$
if $(r_1 == r_2) \wedge (state_{r_1} \neq state_{r_2})$ **then**
    return False
if $(r_1 == r_2 + 1) \wedge (state_{r_1} \neq Update(state_{r_2}, \{Input_{r_1,j}, \sigma_{r_1,j}\}_j))$ **then**
    return False
if $(r_1 > r_2 + 1) \wedge (\exists j, s.t., (pay_{r_1,j}.round \neq pay_{r_2,j}.round) \wedge (pay_{r_1,j}.round \leq r_2))$ **then**
    return False
return True

Algorithm 3: Audit

Input: evidence, bestState

Output: none or punish the custodian

parse evidence as $(state_{r_1}, \sigma_{r_1}, state_{r_2}, \sigma_{r_2})$
for $r \in \{r_1, r_2\}$ **do**
    if $!Vrf_{cus}(state_r, \sigma_r) \wedge (state_r \neq bestState)$ **then**
        $state_r :=\perp$
    if $Vrf_{cus}(state_r, \sigma_r) \wedge !AuditSin(state_r)$ **then**
        punish()
if $(state_{r_1} \neq\perp) \wedge (state_{r_2} \neq\perp) \wedge !AuditDou(state_{r_1}, state_{r_2})$ **then**
    punish()

3 Depart

节点退出通道的流程为：

节点提交退出请求与通道内最新状态 (触发合约的 Depart()) → 其余节点提交退出请求 (触发合约的 Depart())，或提交最新状态 (触发合约的 Record) → 节点完成退出 (触发合约 Resolve())

Contract $\text{Contract}_{\text{MPC}}$

Initialize totalValue := $\sum_j deposit_j$; P := $\{P_j\}_j$; DP := $\emptyset$;

Initialize bestState := (0, $\{(\perp, deposit_j, \emptyset), \perp\}_j$, 0);bestRound := 0;

On **contract input** Depart(state, σ) from P_i at time T:

  if DP == $\emptyset$, set deadline := $T + \Delta$

  DP := DP $\cup\{P_i\}$

  if $state \neq \perp$, Record(state, σ)

  emit EventDeparture(state)

On **contract input** Record($state_r, \sigma$) at time T:

  assert T < deadline

  Audit($state_r, \sigma, bestState, \perp$))

  if r > bestRound, bestState:=$state_r$, bestRound:=r

On **contract input** Resolve() at time T:

  assert (DP $\neq \emptyset$) $\wedge$ (T > deadline)

  bestState = UpdateCon(bestState)

  parse bestState as $(r, \{Input_j, \perp\}_j, F)$

  if (Custodian $\in$ DP) $\wedge(\forall j, s.t., ConPay_j == \emptyset)$ then

    send coins $\$bal_j$ to P_j, \$(F+G) to Custodian

    emit EventClosureH

  else for each party $P_j \in DP \wedge (P_j \neq cus)$

    if $ConPay_j == \emptyset$ then

      send coins $\$bal_j$ to P_j

      P := P $\setminus\{P_j\}$, totalValue -= bal_j

    bestState := $(r+1, \{Input_j, \perp\}_j, F)$, $P_j \in P$

    bestRound := r+1, DP := $\emptyset$

    emit EventResolve(bestState, P, totalValue)

3.1 被调用的 function

Contract $\text{Contract}_{\text{MPC}}$

On **contract input** Audit(evidence):

parse evidence as $(state_{r_1}, \sigma_{r_1}, state_{r_2}, \sigma_{r_2})$

for $r \in \{r_1, r_2\}$,

if $!Vrf_{cus}(state_r, \sigma_r) \wedge (state_r \neq bestState)$, $state_r := \perp$

if $Vrf_{cus}(state_r, \sigma_r) \wedge !AuditSin(state_r)$, punish()

if $(state_{r_1} \neq \perp) \wedge (state_{r_2} \neq \perp) \wedge !AuditDou(state_{r_1}, state_{r_2})$, punish()

Function UpdateCon($state_r$) at time T:

for each cpay

set P_j as $cpay.pr$, P_i as $cpay.pe$

if cpay.time > T

$ConPay_j = ConPay_j \setminus cpay, ConPay_i = ConPay_i \setminus cpay$

if PM.published($cpay.h, cpay.time$), $bal_i += cpay.amt$

else $bal_j += cpay.amt$

return $state_r$

Function Punish()

dbs := (totalValue + G)/ (| P | -1)

send coins \$ dbs to parties in P / $\{Custodian\}$

emit EventClosureM

3.2 全局哈希原像管理合约, the PM contract

Contract $\mathrm{Contract}_{\mathrm{PM}}$

initially timestamp[] is an empty mapping

On **contract input** publish(x) at time T:

  if $\mathcal{H}(x) \notin$ timestamp: then set timestamp[$\mathcal{H}(x)$] := T

contract function published(h, T'):

  return True if $h \in$ timestamp and timestamp[h] $\leq T'$

  return False otherwise