Multi-client Partially Non-Interactive and Instantaneous One-way Payment Channel for Ethereum

September 19, 2018 – DRAFT

Thomas Shababi[1], Joël Gugger[1], and Daniel Lebrecht[1]

TrueLevel SA, Neuchâtel, Switzerland
{tom, joel, d}@truelevel.io

Abstract. Ethereum is a distributed computing platform and operating system featuring smart contract functionality. Transactions are faster than many other blockchains but not instant and each of them cost some "gas". This gas is used to quantify the amount of fee to pay for computation.

Keywords: Crypto-currencies, Ethereum, Payment channels

1 Introduction

Requirements Language The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [1].

2 Multi-clients payment channels

Partially non-interactive multi-clients payment channels are composed of clients $c \in \mathcal{C}$ and one provider $\mathcal{P}$. $\mathcal{C}$ is the set of all clients registered in the multi-clients payment channel. Clients can send money to the provider through the channel but cannot receive through the channel.

A payment channel is by definition a structure composed of two layers of states. The first layer of states is registered on the blockchain, i.e. "on-chain" or contract states $\mu \in M$, and the second layer of states is kept "off-chain" between the participants and the provider, i.e. channel states $\sigma \in \Sigma$. States are sets of deltas. A delta represents an atomic change in a state. Delta changes $\delta \in \Delta$ are used to transition from one state to another.

We denote a main channel $\omega \in \Omega$ and assume the existence of sub-channels. The main channel ω is composed of two layers previously defined (Σ, M).

Each client have a pair of states (μ_c, σ_c) shared with the provider. Only the latest pair is relevant, for each transition the new state in Σ or M is kept. We denote μ_c and σ_c the contract and channel states for a client $c \in \mathcal{C}$. Hereinafter it is implied that all μ, σ correspond to one specific and only one client at any given time, but we forget c to simplify the notation.

Transition from one state to another is denoted with $\rightarrow$, e.g. $\sigma \rightarrow \sigma'$ or $\mu \rightarrow \mu'$. Transitions in Σ depend on particular deltas in Δ_σ noted δ_σ. We denote a transition from $\sigma, \sigma' \in \Sigma$ that depends on $\delta_\sigma \in \Delta_\sigma$ as

$$\sigma \xrightarrow{\delta_\sigma} \sigma'$$

And transitions in M depending on particular deltas in Δ_μ noted δ_μ as

$$\mu \xrightarrow{\delta_\mu} \mu'$$

Transitivity law is applied on transitions. If there exists a transition from a state s to another state s' that depends on t and s' to s'' depending on t', then there exists a transition from s to s'' as

$$s \xrightarrow{t} s' \wedge s' \xrightarrow{t'} s'' \implies s \xrightarrow{t+t'} s''$$

We define predecessor and successor in states Σ and M. If state s is a predecessor of s' we denote them as $s \prec s'$, or s' is a successor of s as $s' \succ s$. We denote the latest transition of a certain type as $\lceil \xrightarrow{t} \rceil$.

Layers are linked together, changes in contract layer affect the channel layer and the channel layer can be dictated, or applied, on the contract layer. We denote the application of σ on μ, i.e. dictate a channel state on the contract layer, also as $\sigma(\mu) = \mu'$. The full transition rule for contract states is defined as

$$\mu \xrightarrow{\delta_\mu \vee \sigma} \mu'$$

and the full transition rule for channel states as

$$\sigma \xrightarrow{\delta_\sigma \vee \mu \rightarrow \mu'} \sigma'$$

We can notice that each transition in contract states M triggers a transition in channel states Σ. States are represented as sets of deltas $\delta \in \Delta$. We define how to compute the resulting state.

$$\begin{aligned}
\mu \xrightarrow{\delta_\mu} \mu' : &\quad \mu' = \mu \cup \{\delta_\mu\} \\
\mu \xrightarrow{\sigma} \mu' : &\quad \mu' = \mu \cup \sigma \\
\sigma \xrightarrow{\delta_\sigma} \sigma' : &\quad \sigma' = \sigma \cup \{\delta_\sigma\} \\
\sigma \xrightarrow{\mu \xrightarrow{\delta_\mu} \mu'} \sigma' : &\quad \sigma' = \sigma \cup \mu' = \sigma \cup \{\delta_\mu\} \\
\sigma \xrightarrow{\mu \xrightarrow{\sigma''} \mu'} \sigma' : &\quad \sigma' = \mu'
\end{aligned}$$

It is worth noting that the last rule $\sigma \xrightarrow{\mu \xrightarrow{\sigma''} \mu'} \sigma'$ has implications on the validity of deltas. Because the resulting channel state σ' is equal to the contract state μ', all deltas δ_σ applied between the dictated state σ'' and the starting state σ, where $\sigma'' \preceq \sigma$, are discarded. If we have

$$\begin{aligned}
&\Sigma : \emptyset \xrightarrow{\emptyset \xrightarrow{\delta_\mu} \mu_1} \sigma_1 \xrightarrow{\delta_\sigma} \sigma_2 \xrightarrow{\delta'_\sigma} \sigma_3 \xrightarrow{\mu_1 \xrightarrow{\sigma_2} \mu_2} \sigma_4 \\
&M : \emptyset \xrightarrow{\delta_\mu} \mu_1 \xrightarrow{\sigma_2} \mu_2
\end{aligned}$$

we can see that a channel state σ can be dictated on a contract state μ if the channel state is a successor of the latest updated state

$$\sigma(\mu) \iff \sigma \succ \sigma'' : \sigma' \lceil \xrightarrow{\mu \xrightarrow{\sigma'''} \mu'} \rceil \sigma''$$

We see that all channel states based on σ_4 will never contain δ'_σ because the dictated channel state used is σ_2 and not σ_3, then as soon as σ_2 is applied σ_3 becomes invalid by the previous rule.

$$\begin{aligned}
\mu_1 &= \emptyset \cup \{\delta_\mu\} & &= \{\delta_\mu\} \\
\sigma_1 &= \emptyset \cup \{\delta_\mu\} & &= \{\delta_\mu\} \\
\sigma_2 &= \{\delta_\mu\} \cup \{\delta_\sigma\} & &= \{\delta_\mu, \delta_\sigma\} \\
\sigma_3 &= \{\delta_\mu, \delta_\sigma\} \cup \{\delta'_\sigma\} & &= \{\delta_\mu, \delta_\sigma, \delta'_\sigma\} \\
\mu_2 &= \{\delta_\mu\} \cup \{\delta_\mu, \delta_\sigma\} & &= \{\delta_\mu, \delta_\sigma\} \\
\sigma_4 &= \{\delta_\mu, \delta_\sigma\} & &= \{\delta_\mu, \delta_\sigma\}
\end{aligned}$$

3 Building blocks

3.1 Contract state variables

We define variables to acces the contract state $\mu \in M$ for a client $c \in \mathcal{C}$.

Current validity index, $\boldsymbol{I_\mu}$ For each client the current index I_μ must be retreivable, $\forall c \in \mathcal{C}, \quad \exists I_\mu \in \mu \geq 0$. The index for a client must start from 0.

Current contract client balance, $\boldsymbol{B_\mu}$ For each client the current contract balance amount must be retrievable, $\forall c \in \mathcal{C}, \quad \exists B_\mu \in \mu \geq 0$. It is worth noting that this current contract balance is not the same as the current channel balance or, in other words, the remaining amount of a client in the channel.

Current provider refund amount, $\boldsymbol{R_\mathcal{P}}$ For each client the current provider refund amount must be retrievable, $\forall c \in \mathcal{C}, \quad \exists R_\mathcal{P} \in \mu$ such that $R_\mathcal{P}$ is the provider owned amount at the moment of the request. The remaining money must go to the client.

3.2 Contract states

Each client $c \in \mathcal{C}$ in the contract is defined by an on-chain state $\mu_c \in M$. States $\mu \in M$ are represented as

$$\mu_c = (I_\mu, B_\mu, R_\mathcal{P})$$

3.3 Channel state variables

We define variables to represent: (i) the lifetime of a single channel in the multi-clients channel architecture, (ii) the total amount deposited for a client over the lifetime, (iii) the total amount sent to the provider over the lifetime, and (iv) the minimal and full available amount for the client.

Channel lifetime It exists one lifetime and only one per element in $\mathcal{C} \times \mathcal{P}$ and begins when the first deposit is made, i.e. one lifetime per client $c \in \mathcal{C}$.

Last observed validity index, $\boldsymbol{I_\sigma}$ The last observed validity index is a copy of contract validity index at the moment where the state is created.

Last observed client balance, $\boldsymbol{B_\sigma}$ Same as the validity index but the client balance.

Total deposit, $\boldsymbol{D}$ Total deposit of a client represents the total amount recieved from the begining of the lifetime. Each top up increases the total deposit.

Total spent, $\boldsymbol{S}$ Total amount sent to the provider by a client represents the sum of all payments since the begining of the lifetime. Each payment increases the total spent.

Client refunded amount, $\boldsymbol{R}$ Total amount refunded to the client since the begining of the lifetime. Each refund can increase the total client refunded amount.

Minimal available amount, $\boldsymbol{A_m}$ Minimal amount available for a client is computed with the latest channel state σ. Without quering the contract state μ it is impossible to know if a bigger amount is now available. The minimal available amount is computed with

$$A_m = D - S - R$$

Client available amount, $\boldsymbol{A}$ The full amount available into a single channel for a client is computed with

$$A = D - S - R + B_\mu - B_\sigma, \quad B_\mu \geq A$$

It is worth noting that $B_\mu - B_\sigma$ is added to the difference of total deposit and total spent in case of new on-chain delta changes.

3.4 Channel states

Each client $c \in \mathcal{C}$ is defined by their channel state $\sigma \in \Sigma$. States $\sigma \in \Sigma$ are composed of: (i) a validity index, (ii) the latest observed on-chain balance of the client, (iii) the total deposit of the client, (iv) the total owned by the provider, and (v) the client total refunded amount. We denote a state $\sigma_c \in \Sigma$ as

$$\sigma_c = (I_\sigma, B_\sigma, D, S, R)$$

3.5 Modifying channel states and contract states with Deltas

Deltas $\delta \in \Delta$ represent external channel capabilities of changing states Σ and contract states M. Along with directives, they encapsulate all the channel capabilities. In this proposed channel architecture capabilities are: send money through the channel from clients to provider, create and top up a channel, settle a channel, and ask and get a refund. In that list, external capabilities are sending money through the channel, creating a channel, and top up a channel.

Payment, $\boldsymbol{\delta_\sigma(a)}$ This delta is used to increase the channel balance of the provider $O(\mathcal{P})$ of a amount from a client c. Payment deltas are sent by clients to the provider. For each payment an acquired settle directive d_s is generated by the client and sent to the provider.

Top up, $\boldsymbol{\delta_\mu(a)}$ This delta is used to increase the contract balance of a client by a amount. Top up deltas are sent by anyone for any specific channel, i.e. client.

Creating channels is a special case of top up, when a channel is top up but does not already exists, the channel is created. The first delta of all channel will always be a top up.

3.6 Dictate channel state on contract layer with Messages & Directives

We define in the previous chapter a transition in contract layer performed with a channel state noted $\mu \xrightarrow{\sigma} \mu'$ or $\sigma(\mu) = \mu'$. These transitions are performed by the intermediary of messages $m \in \mathcal{M}$ and directives d.

Messages $m \in \mathcal{M}$ are an intermediary representation of channel states between the channel layer and contract layer. A message is related to one and only one element in $\mathcal{C} \times \mathcal{P}$ for only one $\sigma \in \Sigma$. Messages are used to derive directives. Directives are the concrete representation of a state σ in the contract layer. The contract layer only "speak" and understand directives.

Thus we will use the message representation in the following chapter to describe contract transitions based on channel state application.

Minimal message A minimal message $m \in \mathcal{M}$ between a client c and the provider $\mathcal{P}$ is composed of four components: (i) a validity index, (ii) the lastest observed balance, (iii) the ajusted total of deposit, and (iv) the total owned by the provider (total spent by the client). We denote a message $m \in \mathcal{M}$ as

$$m = (I_m, B_\sigma, D_m, S)$$

Message derivation Messages are derived from channel states $\varSigma$ as

$$\begin{aligned} \sigma &\implies m \\ (I_\sigma, B_\sigma, D, S, R) &\implies (I_\sigma + 1, B_\sigma, D - R, S) \end{aligned}$$

Directives One message can derives many directives. We define two base directives: (i) one used by the provider and (ii) one used by the client. It is possible to add as many directives as needed in the contract layer. We distinguish two types of directives: (i) self-generated directives, and (ii) acquired directives. Self-generated directives can be generated without external computation and are directly valid in the contract layer. Conversely acquired directive require external computation with the other party (can be client or provider).

Settling and refunding a channel are part of dictate a channel state on the contract layer, and then are performed with directives.

Settle, d_s Directive sent from the client to the provider to validate a payment $\delta_\sigma(a)$. The process of generating the directive can be sumarized as

$$\forall \sigma \xrightarrow{\delta_\sigma(a)} \sigma', \exists d_s : \sigma' \implies m \implies d_s$$

Refund, d_r Self-generated directive by the client used to ask for a refund. The process of generating the directive can be sumarized as

$$\forall \sigma, \exists d_r : \sigma \implies m \implies d_r$$

When a client ask for a refund, the provider has tree choices: (i) doing nothing, (ii) resolve the refund with a matching d_r, or (iii) dispute the refund with a matching d_r. If the provider do nothing the client can take what he asked after some delay, otherwise the provider will close the request by resolve or dispute.

4 State transitions

4.1 Contract state, $\mu \to \mu'$

Contract states $\mu \in M$ can transition from μ to μ' with a message $m \in \mathcal{M}$ (i.e. directive) or contract state changes $\delta_\mu \in \varDelta$.

Settlement $(I_\mu, B_\mu, R_\mathcal{P}) \xrightarrow{m} (I_\mu + 1, B_\mu \downarrow, R_\mathcal{P})$ where

$$m = (I_m, B_\sigma, D_m, S) \mid I_m = I_\mu + 1$$

It is worth noting that with this rule it is not possible to settle a zero amount because the balance amount B_μ must go down. $O(\mathcal{P})$ is the amount owned by the provider at the settlement time. The provider balance is the current contract balance minus the client available funds. Computed as

$$\begin{aligned} O(\mathcal{P}) &= B_\mu - A \\ &= B_\mu - (D_m - S + B_\mu - B_\sigma) \end{aligned}$$

The new balance $B_\mu \downarrow$ is the remaing funds

$$\begin{aligned} B_\mu \downarrow &= A \\ &= D_m - S + B_\mu - B_\sigma \end{aligned}$$

The new client balance is the current contract balance minus the the amount due to the provider.

Refund $(I_\mu, B_\mu, R_\mathcal{P}) \xrightarrow{m} (I_\mu + 1, B_\mu, R'_\mathcal{P})$ where

$$m = (I_m, B_\sigma, D_m, S) \mid I_m = I_\mu + 1$$

It is a full refund if the client available funds in the channel for the client are equal to the current contract balance and a partial refund if the available funds of the client are smaller than the balance

$$R'_\mathcal{P} = \begin{cases} B_\mu - A & \text{if } B_\mu > A \\ 0 & \text{if } B_\mu = A \end{cases}$$

Dispute $(I_\mu, B_\mu, R_\mathcal{P}) \xrightarrow{m} (I_\mu, 0, 0)$ where

$$m = (I_m, B_\sigma, D_m, S) \mid I_m = I_\mu$$

If $O(\mathcal{P}) > R_\mathcal{P}$ the provider wins the dispute and $O(\mathcal{P})$ is transferred to the provider plus some penalty, the rest is sent to the client.

Resolve $(I_\mu, B_\mu, R_\mathcal{P}) \xrightarrow{m} (I_\mu, 0, 0)$ where

$$m = (I_m, B_\sigma, D_m, S) \mid I_m = I_\mu$$

If $O(\mathcal{P}) = R_\mathcal{P}$ the provider accepts the refund. $O(\mathcal{P})$ is transferred to the provider and the rest is sent to the client immediately.

Top up $(I_\mu, B_\mu, R_\mathcal{P}) \xrightarrow{\delta_\mu(a)} (I_\mu, B_\mu + a, R_\mathcal{P})$. Top up increases the contract balance for a client $c \in \mathcal{C}$. Validity index I_μ must not be incremented during the top up.

Invalid settlement $(I_\mu, B_\mu, R_\mathcal{P}) \rightarrow (I_\mu, B_\mu \downarrow, R_\mathcal{P})$. This transition is invalid because the balance is decreased without incrementing the validity index I_μ. The current set of transactions must be invalidated after the settlement.

Invalid top up $(I_\mu, B_\mu, R_\mathcal{P}) \rightarrow (I_\mu + 1, B_\mu \uparrow, R_\mathcal{P})$. Increasing validity index I_μ while increasing the contract balance invalidates the set of transactions I_μ. The current set of transactions must be invalidated only during a settlement.

Invalid validity index $(I_\mu, B_\mu, R_\mathcal{P}) \rightarrow (I_\mu \downarrow, \_, \_)$. Decreasing the validity index I_μ is always invalid. Validity index must not be decreased.

4.2 Channel state, $\sigma \rightarrow \sigma'$

Channel state transitions $(\sigma \xrightarrow{\delta_\sigma \vee \mu \rightarrow \mu'} \sigma')$ are triggered by delta changes $\delta_\sigma \in \Delta$ or by transitions in M, i.e. $\mu \rightarrow \mu'$. The base channel state σ is computed as

$$\sigma = \begin{cases} \emptyset & \text{if no previous state exists} \\ (I_\sigma, B_\sigma, D, S, R) & \text{otherwise} \end{cases}$$

Payment, $\delta_\sigma(a)$. Payments trigger transition in channel states from the base state σ to a destination state σ' noted $\sigma \xrightarrow{\delta_\sigma(a)} \sigma'$. The destination channel state is computed as

$$\sigma' = \begin{cases} \emptyset & \text{if } \sigma = \emptyset \\ (I_\mu, B_\mu, D, S + a, R) & \text{otherwise} \end{cases}$$

Payments cannot be accepted without a previous state. If $\sigma = \emptyset$ then the channel is not initialized yet and must not accept any payments.

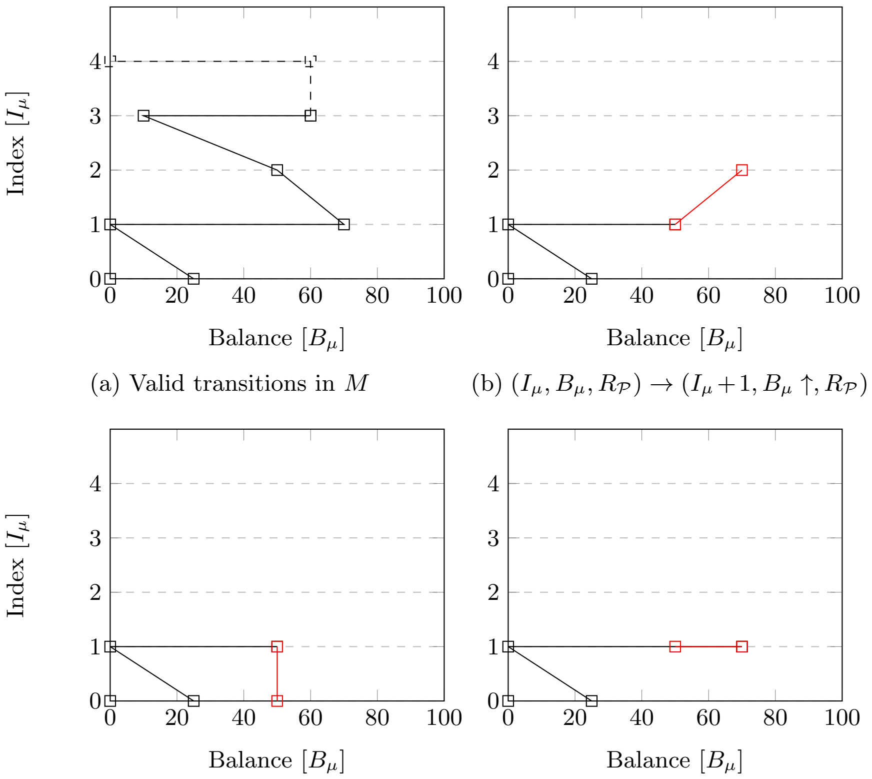
Top up, $\mu \xrightarrow{\delta_\mu(a)} \mu'$. Each top up event modify the contract state μ, then the channel state σ must be updated to keep track of the real balance. The destination channel state is computed as

$$\sigma' = \begin{cases} (0, B_\mu, B_\mu, 0, 0) & \text{if } \sigma = \emptyset \\ (I_\sigma, B_\mu, D + B_\mu - B_\sigma, S, R) & \text{otherwise} \end{cases}$$

where

$$B_\mu - B_\sigma = a$$

It is worth noting that if no previous state exists the channel is initialized and the first state is created.



(a) Valid transitions in M

(b) $(I_\mu, B_\mu, R_\mathcal{P}) \to (I_\mu + 1, B_\mu \uparrow, R_\mathcal{P})$

(c) $(I_\mu, B_\mu, R_\mathcal{P}) \to (I_\mu \downarrow, \_, R_\mathcal{P})$

(d) $(I_\mu, B_\mu, R_\mathcal{P}) \to (I_\mu, B_\mu \downarrow, R_\mathcal{P})$

Fig. 1. State transitions in M, transitions in red are not allowed. Dashed transition are during refund process. The first step is the refund request, the second can be either claim, resolve, or dispute. Figure (c) show validity index decrementation, this is invalid no matter of the resulting balance amount. Figure (d) shows a decreasing balance, red line owerflow the black line.

Settle, $\boldsymbol{\mu \xrightarrow{\sigma''} \mu'}$. Settlements trigger transition in contract state $\mu \to \mu'$ and then the channel state σ must be updated as

$$\sigma' = \begin{cases} \emptyset & \text{if } \sigma = \emptyset \\ (I_\mu, B_\mu, D, S, R) & \text{otherwise} \end{cases}$$

Refund, $\boldsymbol{\mu \xrightarrow{\sigma''} \mu' \mid R_\mathcal{P} \geq 0}$. Clients can ask for refund and modify the contract state then the channel state must be updated as

$$\sigma' = \begin{cases} \emptyset & \text{if } \sigma = \emptyset \\ (I_\mu, B_\sigma, D, S, R) & \text{otherwise} \end{cases}$$

and the state of the channel must enter is “refund mode” until the refund is ended with either a dispute, resolve, or a claim. During this periode client must not send new payments to prevent the provider to successfully dispute the refund (even if updating the channel state to $I_\sigma = I_\mu$ normlay prevent negociating payment for $I_m = I_\mu$).

Dispute, $\boldsymbol{\mu \xrightarrow{\sigma''} \mu'}$. When request refunds are resolved with a dispute the provider can take what he owns during the dispute plus some penalties ρ.

$$\sigma' = \begin{cases} \emptyset & \text{if } \sigma = \emptyset \\ (I_\sigma, B_\mu, D, S + \rho, B_\mu - R_{\mathcal{P}} - \rho) & \text{otherwise} \end{cases}$$

such that

$$D = S + R \wedge B_\mu = 0$$

Resolve & Claim, $\boldsymbol{\mu \xrightarrow{\sigma''} \mu'}$. Provider can accept the request and resolve a refund request by submitting the settle directive d_s or a client can claim his funds after the delay. In both cases the contract state μ changes, then the channel state must be updated as

$$\sigma' = \begin{cases} \emptyset & \text{if } \sigma = \emptyset \\ (I_\sigma, B_\mu, D, S, B_\mu - R_{\mathcal{P}}) & \text{otherwise} \end{cases}$$

such that

$$D = S + R \wedge B_\mu = 0$$

5 Protocol

The protocol describes interactions between clients and the provider and message negotiation. Protocol defines the derivation of directives from messages.

Client actions A client can send money through the channel and refund his money after some time if he submitted the right message m corresponding to the latest channel state σ.

Provider actions The provider can settle channels with his latest message corresponding to the latest channel state. If a client requests a refund, the provider has the choice to dispute it or resolve it.

Anyone actions Anyone can top up a channel for himself or for another client.

	δ	μ	m	σ
	$\delta(a)$	$(I_\mu, B_\mu, R_\mathcal{P})$	(I_m, B_σ, D_m, S)	$(I_\sigma, B_\sigma, D, S, R)$
		$(0, 0, 0)$		$\emptyset$
top up	$\delta_\mu(10)$	$(0, 10, 0)$		$(0, 10, 10, 0, 0)$
payment	$\delta_\sigma(1)$		$(1, 10, 10, 1)$	$(0, 10, 10, 1, 0)$
payment	$\delta_\sigma(1)$		$(1, 10, 10, 2)$	$(0, 10, 10, 2, 0)$
settle		$(1, 8, 0)$		$(1, 8, 10, 2, 0)$
payment	$\delta_\sigma(2)$		$(2, 8, 10, 4)$	$(1, 8, 10, 4, 0)$
payment	$\delta_\sigma(2)$		$(2, 8, 10, 6)$	$(1, 8, 10, 6, 0)$
top up	$\delta_\mu(10)$	$(1, 18, 0)$		$(1, 18, 20, 6, 0)$
payment	$\delta_\sigma(1)$		$(2, 18, 20, 7)$	$(1, 18, 20, 7, 0)$
settle		$(2, 13, 0)$	$(3, 13, 20, 7)$	$(2, 13, 20, 7, 0)$
refund		$(3, 13, 0)$		$(3, 13, 20, 7, 0)$
claim		$(3, 0, 0)$		$(3, 0, 20, 7, 13)$

Table 1. State transitions during channel lifetime

5.1 Payment, $\boldsymbol{\delta_\sigma(a)}$

Payments are negotiated with of one acquire directive for the provider signed by the client.

Settle directive, $\boldsymbol{d_s}$ Settle directive d_s allows the provider to claim immediately, i.e. without any delay, his owned amount for a specific channel. The client must not be able to apply a settle directive on behalf of the provider.

$$
\begin{aligned}
h &= \texttt{keccak256}(\texttt{contract}, c, I_m, B_\sigma, D_m, S) \\
d_s &= \texttt{sign}(\texttt{prefix}, h) \quad \text{with } c \text{ private key}
\end{aligned}
$$

5.2 Refund

Opening a refund is performed with one directive self-generated by the client and signed by himself. The provider must not be able to create directives d_r nor apply a refund directive on behalf of the client.

Refund directive, $\boldsymbol{d_r}$ Refund directive d_r allows the client to claim all the remaining funds and transfer them after some delay if the provider is not able to dispute the claim nor resolves the request. The provider is not able to dispute

	δ	μ	m	σ
	$\delta(a)$	$(I_\mu, B_\mu, R_\mathcal{P})$	(I_m, B_σ, D_m, S)	$(I_\sigma, B_\sigma, D, S, R)$
		$(0, 0, 0)$		$\emptyset$
top up	$\delta_\mu(3)$	$(0, 3, 0)$		$(0, 3, 3, 0, 0)$
top up	$\delta_\mu(2)$	$(0, 5, 0)$		$(0, 5, 5, 0, 0)$
payment	$\delta_\sigma(3)$		$(1, 5, 5, 3)$	$(0, 5, 5, 3, 0)$
settle		$(1, 2, 0)$		$(1, 2, 5, 3, 0)$
payment	$\delta_\sigma(1)$		$(2, 2, 5, 4)$	$(1, 2, 5, 4, 0)$
top up	$\delta_\mu(2)$	$(1, 4, 0)$		$(1, 4, 7, 4, 0)$
settle		$(2, 3, 0)$		$(2, 3, 7, 4, 0)$
payment	$\delta_\sigma(2)$		$(3, 3, 7, 6)$	$(2, 3, 7, 6, 0)$
refund		$(3, 3, 2)$		$(3, 3, 7, 6, 0)$
top up	$\delta_\mu(10)$	$(3, 13, 2)$		$(3, 13, 17, 6, 0)$
resolve		$(3, 0, 0)$		$(3, 0, 17, 6, 11)$
top up	$\delta_\mu(15)$	$(3, 15, 0)$		$(3, 15, 32, 6, 11)$
payment	$\delta_\sigma(7)$		$(4, 15, 21, 13)$	$(3, 15, 32, 13, 11)$
payment	$\delta_\sigma(5)$		$(4, 15, 21, 18)$	$(3, 15, 32, 18, 11)$
top up	$\delta_\mu(3)$	$(3, 18, 0)$		$(3, 15, 35, 18, 11)$
settle		$(4, 6, 0)$		$(4, 6, 35, 18, 11)$

Table 2. Settlement after top up and refund resolved

the claim if the directive is generated from the message derived by the latest channel state in Σ.

$$\begin{aligned} h &= \texttt{keccak256}(\texttt{contract}, c, I_m, O) \\ d_r &= \texttt{sign}(\texttt{prefix}, h) \quad \text{with } c \text{ private key} \end{aligned}$$

with

$$\begin{aligned} O &= B_\mu - A \\ &= B_\mu - (D_m - S + B_\mu - B_\sigma) \end{aligned}$$

A refund request can be closed in three different manners

$$\text{close a refund by} \begin{cases} \text{dispute} & \text{if the provider is able to prove a lie} \\ \text{resolve} & \text{if the provider agrees} \\ \text{claim} & \text{otherwise by the client after delay} \end{cases}$$

Dispute refund Disputes are won if the provider can prove that is owns a payment directive d_s, i.e. client agreed on a later channel state σ', where

$$O' > O : \quad O' \in d_s \wedge O \in d_r$$

with

$$\begin{aligned} O' &= B_\mu - A \\ &= B_\mu - (D_m - S + B_\mu - B_\sigma) \end{aligned}$$

Resolve refund Provider can choose to agree with the request and resolve it directly. By resolving the request the provider will get his remaining funds and send the client available funds A directly to the client. It is worth noting that the provider has no incentive to resolve requests where $S = 0$.

Claim refund The client can claim his funds after the delay if the provider has not disputed nor resolved the request.

5.3 Co-operative Withdraw

Allowing the client to withdraw an amount from the channel is performed with an acquire directive by the provider. Clients must be able to create directives d_w but cannot enforced them. This feature is only available in the case of a honest co-operative provider that accept and enforced the directive. It must not be possible to withdraw more than the client available amount A.

If a client send a withdraw directive to the provider and the provider does not accept it nor enforced it, and if the client spend the available amount A, then the withdraw directive and the latest settle directive are mutualy exclusive because the lack of balance in the channel.

Withdraw directive, d_w Withdraw directive d_w allows the client to withdraw an arbitrary amount and transfer it to an arbitrary location.

$$\begin{aligned} h &= \texttt{keccak256}(\texttt{contract}, c, W, L) \\ d_w &= \texttt{sign}(\texttt{prefix}, h) \quad \text{with } c \text{ private key} \end{aligned}$$

where W is the withdraw amount and L the location where funds are moved.

6 Sub-channel applications

In this chapter we describe how to use sub-channels to build a partially trusted pre-authorized payment layer. Pre-authorized payments in sub-channel ω_p can be merged at any moment into the main channel ω given a partially-trusted model.

Partially-trusted pre-authorized payments model We define in this model of pre-authorized payments the amount of trust required by the client in favour of the provider. We define the trust as

> At any time the client puts at risk at most a pre-authorization of l amount and trusts the provider to take $x \leq l$ amount only when necessary. Client can never loose more than l in the case of a provider misbehaving.

6.1 Pre-authorized payments sub-channel

We first define a unique sub-channel ω_p per client that allows the provider at any moment to take an arbitrary amount $x \leq l$ up to a limit l set by the client. The sub-channel must follow the partially-trusted model defined previously. The client must be able to increase the limit l and merge an arbitrary amount $y \leq l$ at any moment into the main channel ω.

Sub-channel ω_p is composed of a sub-channel state Σ^p and contract state $M^p \subset M$. Sub-channel states are noted $\sigma^p \in \Sigma^p$. We define an increase of the limit l as δ_{σ^p}. The transition between two sub-channel states can be triggered with a delta or because of a merge into the main channel.

We redefine the transition condition for contract states as

$$\mu \xrightarrow{\delta_\mu \vee \sigma \vee \sigma^p} \mu'$$

the redefinition of transition condition for main channel states as

$$\sigma \xrightarrow{\delta_\sigma \vee \sigma^p \vee \mu \rightarrow \mu'} \sigma'$$

and the new transition condition for sub-channel states as

$$\sigma^p \xrightarrow{\delta_{\sigma^p} \vee \sigma \xrightarrow{\sigma^p} \sigma'} \sigma'^p$$

We also define the new transition rules as

$$\mu \xrightarrow{\sigma^p} \mu' : \quad \mu' = \mu \cup \sigma^p$$

$$\sigma \xrightarrow{\sigma^p} \sigma' : \quad \sigma' = \sigma \cup \sigma^p$$

$$\sigma \xrightarrow{\mu \xrightarrow{\sigma^p} \mu'} \sigma' : \quad \sigma' = \sigma \cup \mu'$$

It is worth noting that sub-channel states can be enforced directly on the contract layer or merged into the main channel state, but in both case the main channel state is updated accordingly. We miss two more rule definitions

$$\sigma^p \xrightarrow{\delta_{\sigma^p}} \sigma'^p : \quad \sigma'^p = \sigma^p \cup \{\delta_{\sigma^p}\}$$

$$\sigma^p \xrightarrow{\sigma \xrightarrow{\sigma''^p} \sigma'} \sigma'^p : \quad \sigma'^p = \sigma^p \setminus \sigma''^p$$

When a sub-channel state is merged into the main channel state, the sub-channel deltas are moved into the state channel and removed from the sub-channel.

Sub-channel merge Sub-channel merges move the total limit l (sum of all deltas in sub-channel state) into the main channel and can be defined as

$$\begin{aligned}\Sigma^p : \sigma_1^p &\xrightarrow{\sigma_1 \xrightarrow{\sigma_1^p} \sigma_2} \sigma_2^p \\ \Sigma : \sigma_1 &\xrightarrow{\sigma_1^p} \sigma_2\end{aligned}$$

where σ_2 contains all deltas, i.e. the total limit l. Set's states can be computed with the previous rules given the base states σ_1^p and σ_1 as

$$\begin{aligned}
&\sigma_1^p = \{\delta_{\sigma^p}\} && \\
&\sigma_1 = \{\delta_\sigma\} && \\
&\sigma_2 = \{\delta_\sigma\} \cup \{\delta_{\sigma^p}\} && = \{\delta_\sigma, \delta_{\sigma^p}\} \\
&\sigma_2^p = \{\delta_{\sigma^p}\} \setminus \{\delta_\sigma, \delta_{\sigma^p}\} && = \emptyset
\end{aligned}$$

We say that the main channel state contains a sub-channel state if the sub-channel state has been merged. The rule follows the superset rule

$$\sigma \supset \sigma^p \iff \exists \sigma' \mid \sigma' \xrightarrow{\sigma^p} \sigma$$

Channels' states pairings We define pairings of main channel and sub-channel states. States in pairings can be either non-exclusive $\|$, i.e. both states can be enforced on the contract layer, or mutually exclusive $\perp$, i.e. only one of the two states can be enforced.

We define, as previously shown, four states when a merge happens: $\sigma_1, \sigma_2, \sigma_1^p, \sigma_2^p$. We define three pairings of states (noted $1, 2, 3$) with only one state in channel and sub-channel as

$$\begin{tikzcd}
\sigma_1^p \arrow[r, "\sigma_1 \xrightarrow{\sigma_1^p} \sigma_2"] \arrow[d, dash, dotted, "1"] \arrow[dr, dash, dotted, "2"] & \sigma_2^p \arrow[d, dash, dotted, "3"] \\
\sigma_1 \arrow[r, "\sigma_1^p"] & \sigma_2
\end{tikzcd}$$

To guarantee the partially-trusted model the pairing 1 is non-exclusive and must be valid at any moment but must invalidate the pairing 3. The inverse is also true. The pairing 2 must be mutually exclusive, e.g. if σ_1^p is enforced then $\sigma_2 : \sigma_2 \supset \sigma_1^p$ must be invalid in the contract layer. Otherwise deltas in σ_1^p can be used twice and the partialy-trusted model is broken.

Given the previous case we can derive generic rules

$$\forall (\sigma, \sigma^p) : \sigma \perp \sigma^p \iff \sigma \supset \sigma^p$$

and

$$\forall (\sigma, \sigma^p) : \sigma \parallel \sigma^p \iff \sigma \not\supset \sigma^p$$

The fourth pairing (σ_1, σ_2^p) is valid but does not represent a threat for the clients. Deltas merged in σ_2 are removed in σ_2^p because of the rule

$$\sigma^p \xrightarrow{\sigma \xrightarrow{\sigma''^p} \sigma'} \sigma'^p : \quad \sigma'^p = \sigma^p \setminus \sigma''^p$$

and are not in σ_1. So $\sigma_1 + \sigma_2^p$ have less value than the three other pairings. It is worth noting that the pairing 3 (σ_2, σ_2^p) is equal to $(\sigma_2, \emptyset)$ and then is the same as only enforced σ_2.

Sub-channel partial merge Sub-channel partial merges move an arbitrary amount $y < l$ into the main channel and can be defined as

$$\begin{aligned} &\varSigma^p : \sigma_1^p \xrightarrow{\sigma_1 \xrightarrow{\sigma_1^p} \sigma_2} \sigma_2^p \xrightarrow{\delta'_{\sigma^p}} \sigma_3^p \qquad && \mid l(\delta'_{\sigma^p}) = l - y \\ &\varSigma : \sigma_1 \xrightarrow{\sigma_1^p} \sigma_2 && \mid y(\sigma_1^p) < l \end{aligned}$$

where initial limit $l = \sigma_1^p$ and $\sigma_1^p \perp \sigma_2 \wedge \sigma_3^p \parallel \sigma_2$. The security of a partial merge inherits the security of a merge only if the value of pairing (σ_2, σ_3^p) is greater than or equal to pairing (σ_1, σ_1^p).

6.2 Channel state

We extend the channel state to

$$\sigma_c = (I_\sigma, N_\sigma, B_\sigma, D, S, R)$$

Current merge index, N_σ For each client the current index N_σ must be retreivable, $\forall c \in \mathcal{C}, \quad \exists N_\sigma \in \sigma_c \geq 0$. The merge index for a client must start from 0.

6.3 Sub-channel state

We define the sub-channel state as

$$\sigma_c^p = (I_{\sigma^p}, N_{\sigma^p}, L)$$

Current index, I_{σ^p} For each client the current index I_{σ^p} must be retreivable, $\forall c \in \mathcal{C}, \quad \exists I_{\sigma^p} \in \sigma_c^p \geq 0$. The index for a client must start from 0.

Current merge index, N_{σ^p} For each client the current index N_{σ^p} must be retreivable, $\forall c \in \mathcal{C}, \quad \exists N_{\sigma^p} \in \sigma_c^p \geq 0$. The merge index for a client must start from 0.

Current limit, L For each client the current pre-authorized limit L must be retreivable, $\forall c \in \mathcal{C}, \quad \exists L \in \sigma_c^p \geq 0$.

6.4 Contract state

We extend the contract state to

$$\mu_c = (I_\mu, I_{\mu^p}, N_\mu, N_{\mu^p}, B_\mu, R_\mathcal{P})$$

Current sub-channel index, I_{μ^p} For each client the current index I_{μ^p} must be retreivable, $\forall c \in \mathcal{C}, \quad \exists I_{\mu^p} \in \mu_c \geq 0$. The merge index for a client must start from 0.

Current channel merge index, N_μ For each client the current channel merge index N_μ must be retreivable, $\forall c \in \mathcal{C}, \quad \exists N_\mu \in \mu_c \geq 0$. The channel merge index for a client must start from 0.

Current sub-channel merge index, N_{μ^p} For each client the current sub-channel merge index N_{μ^p} must be retreivable, $\forall c \in \mathcal{C}, \quad \exists N_{\mu^p} \in \mu_c \geq 0$. The sub-channel merge index for a client must start from 0.

6.5 Contract state transition

Validity rules of indexes N and I are defined as follows

$$\begin{aligned}
\mu \xrightarrow{\sigma^p} \mu' : &\quad \sigma^p \parallel \sigma \iff N_{\sigma^p} > N_\mu &\quad | \, \exists \sigma : \mu' \supset \sigma \\
\mu \xrightarrow{\sigma} \mu' : &\quad \sigma^p \parallel \sigma \iff N_\sigma < N_{\mu^p} &\quad | \, \exists \sigma^p : \mu' \supset \sigma^p \\
\sigma \xrightarrow{\sigma^p} \sigma' : &\quad N_\sigma = N_{\sigma^p}
\end{aligned}$$

and

$$I_\mu = I_\sigma \qquad I_{\mu^p} = I_{\sigma^p}$$

References

[1] S. Bradner. *IETF RFC 2119: Key words for use in RFCs to Indicate Requirement Levels.* Tech. rep. 1997.