Reasoning About Multiparty Miniscript

by Russell O'Connor

Blockstream

DRAFT

July 20, 2020

1 Introduction

When constructing scripts holding funds on behalf of multiple parties in the execution of a "smart-contract", different parties in different roles have different constraints that they need the scripts to satisfy. Each party has two kinds of constraints, which I call *authorization* and *control*.TODO: better names? Authorization is a constraint that states the conditions under which one party must be able to redeem the funds. The control constraint states the conditions that one party requires must be met if the funds are spent by anyone.

Put logically, if P is a candidate Miniscript policy for the multiparty "contract" then the authorization constraint for party p, A_p, must entail the policy, and the control constraint for party p, C_p, must be entailed by the policy,

$$A_p \vdash P \text{ and } P \vdash C_p$$

The logical interval $[A_p, C_p]$ specifies a range of Miniscript policies deemed acceptable by one party. In particular we require that $A_p \vdash C_p$ for this party's interval to be non-empty. The intersection of all these intervals for each party dictates a range of Miniscript policies acceptable to all parties.

For a single party, who owns public key $\mathrm{pk}(p)$, their authorization constraint A_p is usually of the form of $\mathrm{pk}(p) \wedge \ldots$ where the $\ldots$ lists the conditions that they accept need to be satisfied in order to redeem the funds. Their control constraint C_p is usually of the form $\mathrm{pk}(p) \vee \ldots$ which states that the funds either the party moved themselves or someone else has moved the funds under some conditions.

Notice that if $A_p = \mathrm{pk}(p) \wedge \ldots \vdash P$ and $P \vdash \mathrm{pk}(p) \vee \ldots = C_p$ then $P = \mathrm{pk}(p)$ lies within the logical interval $[A_p, C_p]$ meaning that the interval is non-empty and that unilateral control over funds is an acceptable Miniscript policy for any given "smart-contract" from any single party's perspective. While each party's interval is usually "self-centered" in this sense, to find a Miniscript policy that satisfies all parties we must take the logical intersection of all parties' intervals.

Finding an acceptable Miniscript policy is simply a matter of choosing any policy within the interval of the disjunction of all parties authorization constraints and the conjunction of all parties control constraints:

$$A = A_{p_1} \vee A_{p_2} \vee \ldots \vee A_{p_n} \vdash P \text{ and } P \vdash C_{p_1} \wedge C_{p_2} \wedge \ldots \wedge C_{p_n} = C$$

Such an interval is non-empty only when $A \vdash C$. In this case it is usually the more liberal policy, C, that is used because it will often contain more spending paths that are still acceptable to every party.

(Note: when each party's contol contraint is of the form $C_p = \mathrm{pk}(p) \vee \ldots$, then the multiparty contraint always contains a $\mathrm{pk}(p_1) \wedge \mathrm{pk}(p_2) \wedge \ldots \wedge \mathrm{pk}(p_n)$ spending path, justifying the taproot design of a μsig aggregated root key.)

2 Examples

2.1 Simple Escrow

In this example, Alice wants to make a payment to Bob in exchange for goods or services. They have contracted a trusted third party, Faythe, to hold the funds in escrow. Upon dispute Faythe will decide whether the payment goes through or is refunded to Alice.

Alice's authorization constraint is that she can redeem the funds whenever Faythe sides with her in a dispute. Alice's control constraint is that if the funds move it is either because she authorized it or Faythe sided with Bob in a dispute.

$$\begin{aligned} A_{\text{Alice}} &:= \text{pk}(\text{Alice}) \wedge \text{pk}(\text{Faythe}) \\ C_{\text{Alice}} &:= \text{pk}(\text{Alice}) \vee (\text{pk}(\text{Bob}) \wedge \text{pk}(\text{Faythe})) \end{aligned}$$

Bob's constraints are symmetric to Alice's. Bob's authorization constrain is that he can redeem the funds whenever Faythe sides with him in a dispute. Bob's control constraint is that if the funds move it is either because he authorized it or Faythe sided with Alice in a dispute.

$$\begin{aligned} A_{\text{Bob}} &:= \text{pk}(\text{Bob}) \wedge \text{pk}(\text{Faythe}) \\ C_{\text{Bob}} &:= \text{pk}(\text{Bob}) \vee (\text{pk}(\text{Alice}) \wedge \text{pk}(\text{Faythe})) \end{aligned}$$

The multiparty authorization constraint is

$$A = (\text{pk}(\text{Alice}) \wedge \text{pk}(\text{Faythe})) \vee (\text{pk}(\text{Bob}) \wedge \text{pk}(\text{Faythe}))$$

The multiparty control constraint is

$$\begin{aligned} C &= (\text{pk}(\text{Alice}) \vee (\text{pk}(\text{Bob}) \wedge \text{pk}(\text{Faythe}))) \wedge (\text{pk}(\text{Bob}) \vee (\text{pk}(\text{Alice}) \wedge \text{pk}(\text{Faythe}))) \\ &= (\text{pk}(\text{Alice}) \wedge \text{pk}(\text{Bob})) \vee (\text{pk}(\text{Alice}) \wedge \text{pk}(\text{Faythe})) \vee (\text{pk}(\text{Bob}) \wedge \text{pk}(\text{Faythe})) \\ &= \text{threshold}(2, [\text{pk}(\text{Alice}), \text{pk}(\text{Bob}), \text{pk}(\text{Faythe})]) \end{aligned}$$

The interval $[A, C]$ is nontrivial. Typically the liberal policy $P := C$ is chosen because it has an extra clause $\text{pk}(\text{Alice}) \wedge \text{pk}(\text{Bob})$ which allows Alice and Bob to redeem the funds together in case of no dispute.

2.2 Hash Time Locked Contracts

In this example, Alice wants to make payment to Bob in exchange for Bob publishing the pre-image of some hash value within a certain time frame.

Alice's authorization constraint is that she can redeem the funds after her timeout, τ_{Alice}. Alice's control constraint is that the funds are either redeemed by herself or the pre-image is revealed.

$$\begin{aligned} A_{\text{Alice}} &:= \text{pk}(\text{Alice}) \wedge \tau_{\text{Alice}} < t \\ C_{\text{Alice}} &:= \text{pk}(\text{Alice}) \vee \text{SHA256}(\text{image}) \end{aligned}$$

Bob's authorization constraint is that he can redeem the funds if he reveals the pre-image. Bob's control constraint is that the funds are either redeemed by himself or his timeout, τ_{Bob} has expired.

$$\begin{aligned} A_{\text{Bob}} &:= \text{pk}(\text{Bob}) \wedge \text{SHA256}(\text{image}) \\ C_{\text{Bob}} &:= \text{pk}(\text{Bob}) \vee \tau_{\text{Bob}} < t \end{aligned}$$

In order to be more general, we have not required that Alice and Bob agree on the same timeout value.

The multiparty authorization constraint is

$$A \;=\; (\mathrm{pk}(\mathrm{Alice}) \wedge \tau_{\mathrm{Alice}} < t) \vee (\mathrm{pk}(\mathrm{Bob}) \wedge \mathrm{SHA256}(\mathrm{image}))$$

The multiparty control constraint is

$$\begin{aligned} C &= (\mathrm{pk}(\mathrm{Alice}) \vee \mathrm{SHA256}(\mathrm{image})) \wedge (\mathrm{pk}(\mathrm{Bob}) \vee \tau_{\mathrm{Bob}} < t) \\ &= (\mathrm{pk}(\mathrm{Alice}) \wedge \mathrm{pk}(\mathrm{Bob})) \vee (\mathrm{pk}(\mathrm{Alice}) \wedge \tau_{\mathrm{Bob}} < t) \vee \\ &\quad\; (\mathrm{pk}(\mathrm{Bob}) \wedge \mathrm{SHA256}(\mathrm{image})) \vee (\mathrm{SHA256}(\mathrm{image}) \wedge \tau_{\mathrm{Bob}} < t) \end{aligned}$$

For the interval $[A, C]$ to be non-empty we require that $\tau_{\mathrm{Bob}} \leq \tau_{\mathrm{Alice}}$.

Again the liberal policy C has more acceptable spending paths than A. However, Miniscript will refuse to compile policy C due to malleability issues with the $(\mathrm{SHA256}(\mathrm{image}) \wedge \tau_{\mathrm{Bob}} < t)$ spending path. In this case the policy

$$P := (\mathrm{pk}(\mathrm{Alice}) \wedge \mathrm{pk}(\mathrm{Bob})) \vee (\mathrm{pk}(\mathrm{Alice}) \wedge \tau_{\mathrm{Bob}} < t) \vee (\mathrm{pk}(\mathrm{Bob}) \wedge \mathrm{SHA256}(\mathrm{image}))$$

is the most liberal acceptable policy that Miniscript will compile. (Notice that τ_{Alice} doesn't occur in the policy, and τ_{Alice}'s only apperance is as a required upper bound on τ_{Bob}.)

3 Verifying Entailment

4 Mixing Height and Time Locks!