

Definitions

```
{-# OPTIONS --safe --without-K --exact-split #-}

open import MLTT.Spartan
open import MLTT.List
open import UF.Subsingletons

open import PredP
open Pred
open ΣPred
open import Lists

module Definitions (Msg : U ) (Secret : U ) where

SxMsg : U
SxMsg = List Secret × (Msg + Secret)

-- We have propositional equality which can be derived from (A → B , B → A)
_↔_ : (A B : W ) → W
A ↔ B = (A → B) × (B → A)
```

At the moment, I consider BSet to not be a proposition. In the future, we might need to have two different definitions, one of it being a proposition.

```
Cm : ∀ V → Pred (Pred SxMsg V) (U ∪ V)
Cm V P = (forall ascrs scrs x → scrs ⊦ ascrs × ascrs ⊦ scrs → P (ascrs , x) ↔ (P (scrs , x)))

BSet : ∀ V → U ∪ V +
BSet V = Σ (Cm V)

-- bset-is-prop : (bs : BSet V) → (forall mp → is-prop (< bs > mp))
-- bset-is-prop bs = bs .pr₂ .pr₁

_symm : (bs : BSet V)
→ (ascrs scrs : List Secret) (x : Msg + Secret) →
( scrs ⊦ ascrs) × (ascrs ⊦ scrs) →
< bs > (ascrs , x) ↔ < bs > (scrs , x)
_symm bs = bs .pr₂
```

Similarly, &PSet might have to be a Proposition in the future, but it increases complexity without any reason at the moment.

```
Cp : ∀ V W → Pred (Pred (2 × (BSet V)) W) (U ∪ V + ∪ W)
Cp V W P = 1

&PSet : ∀ V W → U ∪ V + ∪ W +
&PSet V W = Σ (Cp V W)
```