

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  $\Sigma$ Pred
open import Lists
```

```
module Definitions (Msg :  $\mathcal{U}$ ) (Secret :  $\mathcal{U}'$ ) where
```

```
S×Msg :  $\mathcal{U}'$ 
S×Msg = List Secret × (Msg + Secret)
```

```
-- We have propositional equality which can be derived from (A → B , B → A)
_↔_ : (A B :  $\mathcal{W}$ ) →  $\mathcal{W}'$ 
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 :  $\forall \mathcal{V} \rightarrow \text{Pred (Pred S×Msg } \mathcal{V}) (\mathcal{U} \sqcup \mathcal{V})$ 
Cm  $\mathcal{V}$  P = ( $\forall$  ascrs scrs x → scrs  $\supset$  ascrs × ascrs  $\supset$  scrs → P (ascrs , x) ↔ (P (scrs , x)))
```

```
BSet :  $\forall \mathcal{V} \rightarrow \mathcal{U} \sqcup \mathcal{V} + \cdot$ 
BSet  $\mathcal{V}$  =  $\Sigma$  (Cm  $\mathcal{V}$ )
```

```
-- bset-is-prop : (bs : BSet  $\mathcal{V}$ ) → ( $\forall$  mp → is-prop (< bs > mp))
-- bset-is-prop bs = bs .pr₂ .pr₁
```

```
_symm : (bs : BSet  $\mathcal{V}$ )
→ (ascrs scrs : List Secret) (x : Msg + Secret) →
  (scrs  $\supset$  ascrs) × (ascrs  $\supset$  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 :  $\forall \mathcal{V} \mathcal{W} \rightarrow \text{Pred (Pred (2 × (BSet } \mathcal{V})) \mathcal{W}) (\mathcal{U} \sqcup \mathcal{V} + \sqcup \mathcal{W})$ 
Cp  $\mathcal{V} \mathcal{W}$  P = 1
```

```
&PSet :  $\forall \mathcal{V} \mathcal{W} \rightarrow \mathcal{U} \sqcup \mathcal{V} + \sqcup \mathcal{W} + \cdot$ 
&PSet  $\mathcal{V} \mathcal{W}$  =  $\Sigma$  (Cp  $\mathcal{V} \mathcal{W}$ )
```