Contents

1 cipher Theory

Built: 23 February 2020
Parent Theories: indexedLists, patternMatches

1.1 Datatypes

asymMsg = Ea (’princ pKey) (’message option)

digest = hash (’message option)

pKey = pubK ’princ | privK ’princ

symKey = sym num

symMsg = Es symKey (’message option)

1.2 Definitions

[sign_def]

$\vdash \forall\, pubKey\ \ dgst.$ sign $pubKey\ \ dgst$ = Ea $pubKey$ (SOME $dgst$)

[signVerify_def]

$\vdash \forall\, pubKey\ \ signature\ \ msgContents.$
signVerify $pubKey\ \ signature\ \ msgContents \iff$
(SOME (hash $msgContents$) = deciphP $pubKey\ \ signature$)

1.3 Theorems

[asymMsg_one_one]

$\vdash \forall\, a_0\ \ a_1\ \ a_0'\ \ a_1'.$
(Ea $a_0\ \ a_1$ = Ea $a_0'\ \ a_1'$) $\iff$ (a_0 = a_0') $\wedge$ (a_1 = a_1')

[deciphP_clauses]

$\vdash$ ($\forall\, P\ \ text.$
(deciphP (pubK P) (Ea (privK P) (SOME $text$)) =
SOME $text$) $\wedge$
(deciphP (privK P) (Ea (pubK P) (SOME $text$)) =
SOME $text$)) $\wedge$
($\forall\, k\ \ P\ \ text.$
(deciphP k (Ea (privK P) (SOME $text$)) = SOME $text$) $\iff$
(k = pubK P)) $\wedge$
($\forall\, k\ \ P\ \ text.$

```
      (deciphP k (Ea (pubK P) (SOME text)) = SOME text) ⟺
      (k = privK P)) ∧
   (∀x k2 k1 P2 P1.
      (deciphP (pubK P1) (Ea (pubK P2) (SOME x)) = NONE) ∧
      (deciphP k1 (Ea k2 NONE) = NONE)) ∧
   ∀x P2 P1. deciphP (privK P1) (Ea (privK P2) (SOME x)) = NONE
```

[deciphP_def]

```
⊢ (deciphP key (Ea (privK P) (SOME x)) =
   if key = pubK P then SOME x else NONE) ∧
  (deciphP key (Ea (pubK P) (SOME x)) =
   if key = privK P then SOME x else NONE) ∧
  (deciphP k1 (Ea k2 NONE) = NONE)
```

[deciphP_ind]

```
⊢ ∀P'.
    (∀key P x. P' key (Ea (privK P) (SOME x))) ∧
    (∀key P x. P' key (Ea (pubK P) (SOME x))) ∧
    (∀k1 k2. P' k1 (Ea k2 NONE)) ⇒
    ∀v v1. P' v v1
```

[deciphP_one_one]

```
⊢ (∀P1 P2 text1 text2.
     (deciphP (pubK P1) (Ea (privK P2) (SOME text2)) =
      SOME text1) ⟺ (P1 = P2) ∧ (text1 = text2)) ∧
  (∀P1 P2 text1 text2.
     (deciphP (privK P1) (Ea (pubK P2) (SOME text2)) =
      SOME text1) ⟺ (P1 = P2) ∧ (text1 = text2)) ∧
  (∀p c P msg.
     (deciphP (pubK P) (Ea p c) = SOME msg) ⟺
     (p = privK P) ∧ (c = SOME msg)) ∧
  (∀enMsg P msg.
     (deciphP (pubK P) enMsg = SOME msg) ⟺
     (enMsg = Ea (privK P) (SOME msg))) ∧
  (∀p c P msg.
     (deciphP (privK P) (Ea p c) = SOME msg) ⟺
     (p = pubK P) ∧ (c = SOME msg)) ∧
  ∀enMsg P msg.
    (deciphP (privK P) enMsg = SOME msg) ⟺
    (enMsg = Ea (pubK P) (SOME msg))
```

[deciphS_clauses]

```
⊢ (∀k text. deciphS k (Es k (SOME text)) = SOME text) ∧
  (∀k1 k2 text.
```

```
      (deciphS k1 (Es k2 (SOME text)) = SOME text) ⟺
      (k1 = k2)) ∧
   (∀k1 k2 text.
      (deciphS k1 (Es k2 (SOME text)) = NONE) ⟺ k1 ≠ k2) ∧
   ∀k1 k2. deciphS k1 (Es k2 NONE) = NONE
```

[deciphS_def]

```
⊢ (deciphS k1 (Es k2 (SOME x)) =
    if k1 = k2 then SOME x else NONE) ∧
   (deciphS k1 (Es k2 NONE) = NONE)
```

[deciphS_ind]

```
⊢ ∀P.
    (∀k1 k2 x. P k1 (Es k2 (SOME x))) ∧
    (∀k1 k2. P k1 (Es k2 NONE)) ⇒
    ∀v v1. P v v1
```

[deciphS_one_one]

```
⊢ (∀k1 k2 text1 text2.
      (deciphS k1 (Es k2 (SOME text2)) = SOME text1) ⟺
      (k1 = k2) ∧ (text1 = text2)) ∧
   ∀enMsg text key.
     (deciphS key enMsg = SOME text) ⟺
     (enMsg = Es key (SOME text))
```

[digest_one_one]

```
⊢ ∀a a'. (hash a = hash a') ⟺ (a = a')
```

[option_distinct]

```
⊢ ∀x. NONE ≠ SOME x
```

[option_one_one]

```
⊢ ∀x y. (SOME x = SOME y) ⟺ (x = y)
```

[pKey_distinct_clauses]

```
⊢ (∀a' a. pubK a ≠ privK a') ∧ ∀a' a. privK a' ≠ pubK a
```

[pKey_one_one]

```
⊢ (∀a a'. (pubK a = pubK a') ⟺ (a = a')) ∧
  ∀a a'. (privK a = privK a') ⟺ (a = a')
```

[sign_one_one]

$\vdash \forall pubKey_1\ pubKey_2\ m_1\ m_2.$
$\texttt{(sign}\ pubKey_1\ \texttt{(hash}\ m_1\texttt{) = sign}\ pubKey_2\ \texttt{(hash}\ m_2\texttt{))} \iff$
$\texttt{(}pubKey_1\ \texttt{=}\ pubKey_2\texttt{)} \wedge \texttt{(}m_1\ \texttt{=}\ m_2\texttt{)}$

[signVerify_one_one]

$\vdash (\forall P\ m_1\ m_2.$
$\texttt{signVerify (pubK}\ P\texttt{) (Ea (privK}\ P\texttt{) (SOME (hash (SOME}\ m_1\texttt{))))}$
$\texttt{(SOME}\ m_2\texttt{)} \iff \texttt{(}m_1\ \texttt{=}\ m_2\texttt{))} \wedge$
$(\forall signature\ P\ text.$
$\texttt{signVerify (pubK}\ P\texttt{)}\ signature\ \texttt{(SOME}\ text\texttt{)} \iff$
$\texttt{(}signature\ \texttt{= sign (privK}\ P\texttt{) (hash (SOME}\ text\texttt{))))} \wedge$
$\forall text_2\ text_1\ P_2\ P_1.$
$\texttt{signVerify (pubK}\ P_1\texttt{) (sign (privK}\ P_2\texttt{) (hash (SOME}\ text_2\texttt{)))}$
$\texttt{(SOME}\ text_1\texttt{)} \iff \texttt{(}P_1\ \texttt{=}\ P_2\texttt{)} \wedge \texttt{(}text_1\ \texttt{=}\ text_2\texttt{)}$

[signVerifyOK]

$\vdash \forall P\ msg.$
$\texttt{signVerify (pubK}\ P\texttt{) (sign (privK}\ P\texttt{) (hash (SOME}\ msg\texttt{)))}$
$\texttt{(SOME}\ msg\texttt{)}$

[symKey_one_one]

$\vdash \forall a\ a'.\ \texttt{(sym}\ a\ \texttt{= sym}\ a'\texttt{)} \iff \texttt{(}a\ \texttt{=}\ a'\texttt{)}$

[symMsg_one_one]

$\vdash \forall a_0\ a_1\ a_0'\ a_1'.$
$\texttt{(Es}\ a_0\ a_1\ \texttt{= Es}\ a_0'\ a_1'\texttt{)} \iff \texttt{(}a_0\ \texttt{=}\ a_0'\texttt{)} \wedge \texttt{(}a_1\ \texttt{=}\ a_1'\texttt{)}$

2 cryptoExercises Theory

Built: 23 February 2020
Parent Theories: cipher, string

2.1 Theorems

[exercise15_6_1a_thm]

$\vdash \forall key\ enMsg\ message.$
$\texttt{(deciphS}\ key\ enMsg\ \texttt{= SOME}\ message\texttt{)} \iff$
$\texttt{(}enMsg\ \texttt{= Es}\ key\ \texttt{(SOME}\ message\texttt{))}$

[exercise15_6_1b_thm]

```
⊢ ∀ keyAlice k text.
    (deciphS keyAlice (Es k (SOME text)) =
     SOME "This is from Alice") ⟺
    (k = keyAlice) ∧ (text = "This is from Alice")
```

[exercise15_6_2a_thm]

```
⊢ ∀ P message.
    (deciphP (pubK P) enMsg = SOME message) ⟺
    (enMsg = Ea (privK P) (SOME message))
```

[exercise15_6_2b_thm]

```
⊢ ∀ key text.
    (deciphP (pubK Alice) (Ea key (SOME text)) =
     SOME "This is from Alice") ⟺
    (key = privK Alice) ∧ (text = "This is from Alice")
```

[exercise15_6_3_thm]

```
⊢ ∀ signature.
    signVerify (pubK Alice) signature
      (SOME "This is from Alice") ⟺
    (signature =
     sign (privK Alice) (hash (SOME "This is from Alice")))
```

Index