

Compositional properties of crypto-based components

Maria Spichkova

March 19, 2025

Abstract

This paper presents an Isabelle/HOL [1] set of theories which allows to specify crypto-based components and to verify their composition properties wrt. cryptographic aspects. We introduce a formalisation of the security property of data secrecy, the corresponding definitions and proofs. A part of these definitions is based on [3].

Please note that here we import the Isabelle/HOL theory ListExtras.thy, presented in [2].

Contents

1	Auxiliary data types	2
2	Correctness of the relations between sets of Input/Output channels	2
3	Secrecy: Definitions and properties	4
4	Local Secrets of a component	17
5	Knowledge of Keys and Secrets	22

1 Auxiliary data types

```
theory Secrecy-types
imports Main
begin
```

— We assume disjoint sets: Data of data values,
— Secrets of unguessable values, Keys - set of cryptographic keys.
— Based on these sets, we specify the sets EncType of encryptors that may be
— used for encryption or decryption, and Expression of expression items.
— The specification (component) identifiers should be listed in the set specID,
— the channel indentifiers should be listed in the set chanID.

```
datatype Keys = CKey | CKeyP | SKey | SKeyP | genKey
datatype Secrets = secretD | N | NA
type-synonym Var = nat
type-synonym Data = nat
datatype KS = kKS Keys | sKS Secrets
datatype EncType = kEnc Keys | vEnc Var
datatype specID = sComp1 | sComp2 | sComp3 | sComp4
datatype Expression = kE Keys | sE Secrets | dE Data | idE specID
datatype chanID = ch1 | ch2 | ch3 | ch4
```

```
primrec Expression2KSL:: Expression list  $\Rightarrow$  KS list
where
```

```
Expression2KSL [] = [] |
Expression2KSL (x#xs) =
  ((case x of (kE m)  $\Rightarrow$  [kKS m]
           | (sE m)  $\Rightarrow$  [sKS m]
           | (dE m)  $\Rightarrow$  []
           | (idE m)  $\Rightarrow$  []) @ Expression2KSL xs)
```

```
primrec KS2Expression:: KS  $\Rightarrow$  Expression
where
```

```
KS2Expression (kKS m) = (kE m) |
KS2Expression (sKS m) = (sE m)
```

```
end
```

2 Correctness of the relations between sets of Input/Output channels

```
theory inout
imports Secrecy-types
begin
```

```
consts
  subcomponents :: specID  $\Rightarrow$  specID set
```

— Mappings, defining sets of input, local, and output channels
— of a component

consts

$ins :: specID \Rightarrow chanID\ set$
 $loc :: specID \Rightarrow chanID\ set$
 $out :: specID \Rightarrow chanID\ set$

— Predicate insuring the correct mapping from the component identifier
— to the set of input channels of a component

definition

$inStream :: specID \Rightarrow chanID\ set \Rightarrow bool$

where

$inStream\ x\ y \equiv (ins\ x = y)$

— Predicate insuring the correct mapping from the component identifier
— to the set of local channels of a component

definition

$locStream :: specID \Rightarrow chanID\ set \Rightarrow bool$

where

$locStream\ x\ y \equiv (loc\ x = y)$

— Predicate insuring the correct mapping from the component identifier
— to the set of output channels of a component

definition

$outStream :: specID \Rightarrow chanID\ set \Rightarrow bool$

where

$outStream\ x\ y \equiv (out\ x = y)$

— Predicate insuring the correct relations between
— to the set of input, output and local channels of a component

definition

$correctInOutLoc :: specID \Rightarrow bool$

where

$correctInOutLoc\ x \equiv$
 $(ins\ x) \cap (out\ x) = \{\}$
 $\wedge (ins\ x) \cap (loc\ x) = \{\}$
 $\wedge (loc\ x) \cap (out\ x) = \{\}$

— Predicate insuring the correct relations between
— sets of input channels within a composed component

definition

$correctCompositionIn :: specID \Rightarrow bool$

where

$correctCompositionIn\ x \equiv$
 $(ins\ x) = (\bigcup (ins\ ' (subcomponents\ x)) - (loc\ x))$
 $\wedge (ins\ x) \cap (\bigcup (out\ ' (subcomponents\ x))) = \{\}$

— Predicate insuring the correct relations between

— sets of output channels within a composed component

definition

correctCompositionOut :: *specID* \Rightarrow *bool*

where

correctCompositionOut *x* \equiv

$(out\ x) = (\bigcup (out\ ' (subcomponents\ x)) - (loc\ x))$

$\wedge (out\ x) \cap (\bigcup (ins\ ' (subcomponents\ x))) = \{\}$

— Predicate insuring the correct relations between

— sets of local channels within a composed component

definition

correctCompositionLoc :: *specID* \Rightarrow *bool*

where

correctCompositionLoc *x* \equiv

$(loc\ x) = \bigcup (ins\ ' (subcomponents\ x))$

$\cap \bigcup (out\ ' (subcomponents\ x))$

— If a component is an elementary one (has no subcomponents)

— its set of local channels should be empty

lemma *subcomponents-loc*:

assumes *correctCompositionLoc* *x*

and *subcomponents* *x* = $\{\}$

shows *loc* *x* = $\{\}$

<proof>

end

3 Secrecy: Definitions and properties

theory *Secrecy*

imports *Secrecy-types inout ListExtras*

begin

— Encryption, decryption, signature creation and signature verification functions

— For these functions we define only their signatures and general axioms,

— because in order to reason effectively, we view them as abstract functions and

— abstract from their implementation details

consts

Enc :: *Keys* \Rightarrow *Expression list* \Rightarrow *Expression list*

Decr :: *Keys* \Rightarrow *Expression list* \Rightarrow *Expression list*

Sign :: *Keys* \Rightarrow *Expression list* \Rightarrow *Expression list*

Ext :: *Keys* \Rightarrow *Expression list* \Rightarrow *Expression list*

— Axioms on relations between encryption and decryption keys

axiomatization

EncrDecrKeys :: *Keys* \Rightarrow *Keys* \Rightarrow *bool*

where

ExtSign:

EncrDecrKeys *K1 K2* \longrightarrow (*Ext* *K1* (*Sign* *K2* *E*)) = *E* **and**

DecrEnc:

$$\text{EncrDecrKeys } K1 \ K2 \longrightarrow (\text{Decr } K2 \ (\text{Enc } K1 \ E)) = E$$

— Set of private keys of a component

consts

$$\text{specKeys} :: \text{specID} \Rightarrow \text{Keys set}$$

— Set of unguessable values used by a component

consts

$$\text{specSecrets} :: \text{specID} \Rightarrow \text{Secrets set}$$

— Join set of private keys and unguessable values used by a component

definition

$$\text{specKeysSecrets} :: \text{specID} \Rightarrow \text{KS set}$$

where

$$\text{specKeysSecrets } C \equiv$$

$$\{y . \exists x. y = (kKS \ x) \wedge (x \in (\text{specKeys } C))\} \cup$$

$$\{z . \exists s. z = (sKS \ s) \wedge (s \in (\text{specSecrets } C))\}$$

— Predicate defining that a list of expression items does not contain

— any private key or unguessable value used by a component

definition

$$\text{notSpecKeysSecretsExpr} :: \text{specID} \Rightarrow \text{Expression list} \Rightarrow \text{bool}$$

where

$$\text{notSpecKeysSecretsExpr } P \ e \equiv$$

$$(\forall x. (kE \ x) \ \text{mem } e \longrightarrow (kKS \ x) \notin \text{specKeysSecrets } P) \wedge$$

$$(\forall y. (sE \ y) \ \text{mem } e \longrightarrow (sKS \ y) \notin \text{specKeysSecrets } P)$$

— If a component is a composite one, the set of its private keys

— is a union of the subcomponents' sets of the private keys

definition

$$\text{correctCompositionKeys} :: \text{specID} \Rightarrow \text{bool}$$

where

$$\text{correctCompositionKeys } x \equiv$$

$$\text{subcomponents } x \neq \{\} \longrightarrow$$

$$\text{specKeys } x = \bigcup (\text{specKeys } ` (\text{subcomponents } x))$$

— If a component is a composite one, the set of its unguessable values

— is a union of the subcomponents' sets of the unguessable values

definition

$$\text{correctCompositionSecrets} :: \text{specID} \Rightarrow \text{bool}$$

where

$$\text{correctCompositionSecrets } x \equiv$$

$$\text{subcomponents } x \neq \{\} \longrightarrow$$

$$\text{specSecrets } x = \bigcup (\text{specSecrets } ` (\text{subcomponents } x))$$

— If a component is a composite one, the set of its private keys and

— unguessable values is a union of the corresponding sets of its subcomponents

definition

$$\text{correctCompositionKS} :: \text{specID} \Rightarrow \text{bool}$$

where

$correctCompositionKS\ x \equiv$
 $subcomponents\ x \neq \{\} \longrightarrow$
 $specKeysSecrets\ x = \bigcup (specKeysSecrets\ ' (subcomponents\ x))$

- Predicate defining set of correctness properties of the component's
- interface and relations on its private keys and unguessable values

definition

$correctComponentSecrecy :: specID \Rightarrow bool$

where

$correctComponentSecrecy\ x \equiv$
 $correctCompositionKS\ x \wedge$
 $correctCompositionSecrets\ x \wedge$
 $correctCompositionKeys\ x \wedge$
 $correctCompositionLoc\ x \wedge$
 $correctCompositionIn\ x \wedge$
 $correctCompositionOut\ x \wedge$
 $correctInOutLoc\ x$

- Predicate $exprChannel\ I\ E$ defines whether the expression item E can be sent via the channel I

consts

$exprChannel :: chanID \Rightarrow Expression \Rightarrow bool$

- Predicate $eoutM\ sP\ M\ E$ defines whether the component sP may eventually
- output an expression E if there exists a time interval t of
- an output channel which contains this expression E

definition

$eout :: specID \Rightarrow Expression \Rightarrow bool$

where

$eout\ sP\ E \equiv$
 $\exists (ch :: chanID). ((ch \in (out\ sP)) \wedge (exprChannel\ ch\ E))$

- Predicate $eout\ sP\ E$ defines whether the component sP may eventually
- output an expression E via subset of channels M ,
- which is a subset of output channels of sP ,
- and if there exists a time interval t of
- an output channel which contains this expression E

definition

$eoutM :: specID \Rightarrow chanID\ set \Rightarrow Expression \Rightarrow bool$

where

$eoutM\ sP\ M\ E \equiv$
 $\exists (ch :: chanID). ((ch \in (out\ sP)) \wedge (ch \in M) \wedge (exprChannel\ ch\ E))$

- Predicate $ineM\ sP\ M\ E$ defines whether a component sP may eventually
- get an expression E if there exists a time interval t of
- an input stream which contains this expression E

definition

$ine :: specID \Rightarrow Expression \Rightarrow bool$

where

$ine\ sP\ E \equiv$
 $\exists (ch :: chanID). ((ch \in (ins\ sP)) \wedge (exprChannel\ ch\ E))$

- Predicate $ine\ sP\ E$ defines whether a component sP may eventually
- get an expression E via subset of channels M ,
- which is a subset of input channels of sP ,
- and if there exists a time interval t of
- an input stream which contains this expression E

definition

$ineM :: specID \Rightarrow chanID\ set \Rightarrow Expression \Rightarrow bool$

where

$ineM\ sP\ M\ E \equiv$
 $\exists (ch :: chanID). ((ch \in (ins\ sP)) \wedge (ch \in M) \wedge (exprChannel\ ch\ E))$

- This predicate defines whether an input channel ch of a component sP
- is the only one input channel of this component
- via which it may eventually output an expression E

definition

$out-exprChannelSingle :: specID \Rightarrow chanID \Rightarrow Expression \Rightarrow bool$

where

$out-exprChannelSingle\ sP\ ch\ E \equiv$
 $(ch \in (out\ sP)) \wedge$
 $(exprChannel\ ch\ E) \wedge$
 $(\forall (x :: chanID) (t :: nat). ((x \in (out\ sP)) \wedge (x \neq ch) \longrightarrow \neg exprChannel\ x\ E))$

- This predicate yields true if only the channels from the set $chSet$,
- which is a subset of input channels of the component sP ,
- may eventually output an expression E

definition

$out-exprChannelSet :: specID \Rightarrow chanID\ set \Rightarrow Expression \Rightarrow bool$

where

$out-exprChannelSet\ sP\ chSet\ E \equiv$
 $((\forall (x :: chanID). ((x \in chSet) \longrightarrow ((x \in (out\ sP)) \wedge (exprChannel\ x\ E))))$
 \wedge
 $(\forall (x :: chanID). ((x \notin chSet) \wedge (x \in (out\ sP)) \longrightarrow \neg exprChannel\ x\ E)))$

- This predicate defines whether
- an input channel ch of a component sP is the only one input channel
- of this component via which it may eventually get an expression E

definition

$ine-exprChannelSingle :: specID \Rightarrow chanID \Rightarrow Expression \Rightarrow bool$

where

$ine-exprChannelSingle\ sP\ ch\ E \equiv$
 $(ch \in (ins\ sP)) \wedge$
 $(exprChannel\ ch\ E) \wedge$
 $(\forall (x :: chanID) (t :: nat). ((x \in (ins\ sP)) \wedge (x \neq ch) \longrightarrow \neg exprChannel\ x\ E))$

- This predicate yields true if the component sP may eventually

- get an expression E only via the channels from the set $chSet$,
- which is a subset of input channels of sP

definition

$ine-exprChannelSet :: specID \Rightarrow chanID\ set \Rightarrow Expression \Rightarrow bool$

where

$ine-exprChannelSet\ sP\ chSet\ E \equiv$
 $((\forall (x :: chanID). ((x \in chSet) \longrightarrow ((x \in (ins\ sP)) \wedge (exprChannel\ x\ E))))$
 \wedge
 $(\forall (x :: chanID). ((x \notin chSet) \wedge (x \in (ins\ sP)) \longrightarrow \neg exprChannel\ x\ E)))$

- If a list of expression items does not contain any private key
- or unguessable value of a component P , then the first element
- of the list is neither a private key nor unguessable value of P

lemma *notSpecKeysSecretsExpr-L1:*

assumes *notSpecKeysSecretsExpr* P ($a \# l$)

shows *notSpecKeysSecretsExpr* P [a]

<proof>

lemma *notSpecKeysSecretsExpr-L2:*

assumes *notSpecKeysSecretsExpr* P ($a \# l$)

shows *notSpecKeysSecretsExpr* P l

<proof>

lemma *correctCompositionIn-L1:*

assumes *subcomponents* $PQ = \{P, Q\}$

and *correctCompositionIn* PQ

and $ch \notin loc\ PQ$

and $ch \in ins\ P$

shows $ch \in ins\ PQ$

<proof>

lemma *correctCompositionIn-L2:*

assumes *subcomponents* $PQ = \{P, Q\}$

and *correctCompositionIn* PQ

and $ch \in ins\ PQ$

shows $(ch \in ins\ P) \vee (ch \in ins\ Q)$

<proof>

lemma *ineM-L1:*

assumes $ch \in M$

and $ch \in ins\ P$

and *exprChannel* $ch\ E$

shows *ineM* $P\ M\ E$

<proof>

lemma *ineM-ine:*

assumes *ineM* $P\ M\ E$

shows *ine* $P\ E$

<proof>

lemma *not-ine-ineM:*

assumes $\neg ine\ P\ E$

shows $\neg \text{ineM } P \ M \ E$
<proof>

lemma *eoutM-eout*:
assumes *eoutM* $P \ M \ E$
shows *eout* $P \ E$
<proof>

lemma *not-eout-eoutM*:
assumes $\neg \text{eout } P \ E$
shows $\neg \text{eoutM } P \ M \ E$
<proof>

lemma *correctCompositionKeys-subcomp1*:
assumes *correctCompositionKeys* C
 and $x \in \text{subcomponents } C$
 and $xb \in \text{specKeys } C$
shows $\exists x \in \text{subcomponents } C. (xb \in \text{specKeys } x)$
<proof>

lemma *correctCompositionSecrets-subcomp1*:
assumes *correctCompositionSecrets* C
 and $x \in \text{subcomponents } C$
 and $s \in \text{specSecrets } C$
shows $\exists x \in \text{subcomponents } C. (s \in \text{specSecrets } x)$
<proof>

lemma *correctCompositionKeys-subcomp2*:
assumes *correctCompositionKeys* C
 and $xb \in \text{subcomponents } C$
 and $xc \in \text{specKeys } xb$
shows $xc \in \text{specKeys } C$
<proof>

lemma *correctCompositionSecrets-subcomp2*:
assumes *correctCompositionSecrets* C
 and $xb \in \text{subcomponents } C$
 and $xc \in \text{specSecrets } xb$
shows $xc \in \text{specSecrets } C$
<proof>

lemma *correctCompKS-Keys*:
assumes *correctCompositionKS* C
shows *correctCompositionKeys* C
<proof>

lemma *correctCompKS-Secrets*:
assumes *correctCompositionKS* C
shows *correctCompositionSecrets* C

<proof>

lemma *correctCompKS-KeysSecrets:*
assumes *correctCompositionKeys C*
 and *correctCompositionSecrets C*
shows *correctCompositionKS C*
<proof>

lemma *correctCompositionKS-subcomp1:*
assumes *correctCompositionKS C*
 and *h1:x ∈ subcomponents C*
 and *xa ∈ specKeys C*
shows $\exists y \in \text{subcomponents } C. (xa \in \text{specKeys } y)$
<proof>

lemma *correctCompositionKS-subcomp2:*
assumes *correctCompositionKS C*
 and *h1:x ∈ subcomponents C*
 and *xa ∈ specSecrets C*
shows $\exists y \in \text{subcomponents } C. xa \in \text{specSecrets } y$
<proof>

lemma *correctCompositionKS-subcomp3:*
assumes *correctCompositionKS C*
 and *x ∈ subcomponents C*
 and *xa ∈ specKeys x*
shows *xa ∈ specKeys C*
<proof>

lemma *correctCompositionKS-subcomp4:*
assumes *correctCompositionKS C*
 and *x ∈ subcomponents C*
 and *xa ∈ specSecrets x*
shows *xa ∈ specSecrets C*
<proof>

lemma *correctCompositionKS-PQ:*
assumes *subcomponents PQ = {P, Q}*
 and *correctCompositionKS PQ*
 and *ks ∈ specKeysSecrets PQ*
shows $ks \in \text{specKeysSecrets } P \vee ks \in \text{specKeysSecrets } Q$
<proof>

lemma *correctCompositionKS-neg1:*
assumes *subcomponents PQ = {P, Q}*
 and *correctCompositionKS PQ*
 and $ks \notin \text{specKeysSecrets } P$
 and $ks \notin \text{specKeysSecrets } Q$
shows $ks \notin \text{specKeysSecrets } PQ$

<proof>

lemma *correctCompositionKS-negP*:
assumes *subcomponents PQ = {P, Q}*
 and *correctCompositionKS PQ*
 and *ks ∉ specKeysSecrets PQ*
shows *ks ∉ specKeysSecrets P*
<proof>

lemma *correctCompositionKS-negQ*:
assumes *subcomponents PQ = {P, Q}*
 and *correctCompositionKS PQ*
 and *ks ∉ specKeysSecrets PQ*
shows *ks ∉ specKeysSecrets Q*
<proof>

lemma *out-exprChannelSingle-Set*:
assumes *out-exprChannelSingle P ch E*
shows *out-exprChannelSet P {ch} E*
<proof>

lemma *out-exprChannelSet-Single*:
assumes *out-exprChannelSet P {ch} E*
shows *out-exprChannelSingle P ch E*
<proof>

lemma *ine-exprChannelSingle-Set*:
assumes *ine-exprChannelSingle P ch E*
 shows *ine-exprChannelSet P {ch} E*
<proof>

lemma *ine-exprChannelSet-Single*:
assumes *ine-exprChannelSet P {ch} E*
shows *ine-exprChannelSingle P ch E*
<proof>

lemma *ine-ins-neg1*:
assumes \neg *ine P m*
 and *exprChannel x m*
shows *x ∉ ins P*
<proof>

theorem *TBtheorem1a*:
assumes *ine PQ E*
 and *subcomponents PQ = {P, Q}*
 and *correctCompositionIn PQ*
shows *ine P E ∨ ine Q E*
<proof>

theorem *TBtheorem1b*:
assumes *ineM PQ M E*
and *subcomponents PQ = {P,Q}*
and *correctCompositionIn PQ*
shows $ineM P M E \vee ineM Q M E$
<proof>

theorem *TBtheorem2a*:
assumes *eout PQ E*
and *subcomponents PQ = {P,Q}*
and *correctCompositionOut PQ*
shows $eout P E \vee eout Q E$
<proof>

theorem *TBtheorem2b*:
assumes *eoutM PQ M E*
and *subcomponents PQ = {P,Q}*
and *correctCompositionOut PQ*
shows $eoutM P M E \vee eoutM Q M E$
<proof>

lemma *correctCompositionIn-prop1*:
assumes *subcomponents PQ = {P,Q}*
and *correctCompositionIn PQ*
and $x \in (ins PQ)$
shows $(x \in (ins P)) \vee (x \in (ins Q))$
<proof>

lemma *correctCompositionOut-prop1*:
assumes *subcomponents PQ = {P,Q}*
and *correctCompositionOut PQ*
and $x \in (out PQ)$
shows $(x \in (out P)) \vee (x \in (out Q))$
<proof>

theorem *TBtheorem3a*:
assumes $\neg (ine P E)$
and $\neg (ine Q E)$
and *subcomponents PQ = {P,Q}*
and *correctCompositionIn PQ*
shows $\neg (ine PQ E)$
<proof>

theorem *TBlemma3b*:
assumes $h1: \neg (ineM P M E)$
and $h2: \neg (ineM Q M E)$
and *subPQ:subcomponents PQ = {P,Q}*
and *cCompI:correctCompositionIn PQ*
and $chM:ch \in M$

and $chPQ: ch \in ins\ PQ$
and $eCh: exprChannel\ ch\ E$
shows $False$
 $\langle proof \rangle$

theorem $TBtheorem3b$:
assumes $\neg (ineM\ P\ M\ E)$
and $\neg (ineM\ Q\ M\ E)$
and $subcomponents\ PQ = \{P, Q\}$
and $correctCompositionIn\ PQ$
shows $\neg (ineM\ PQ\ M\ E)$
 $\langle proof \rangle$

theorem $TBtheorem4a-empty$:
assumes $(ine\ P\ E) \vee (ine\ Q\ E)$
and $subcomponents\ PQ = \{P, Q\}$
and $correctCompositionIn\ PQ$
and $loc\ PQ = \{\}$
shows $ine\ PQ\ E$
 $\langle proof \rangle$

theorem $TBtheorem4a-P$:
assumes $ine\ P\ E$
and $subcomponents\ PQ = \{P, Q\}$
and $correctCompositionIn\ PQ$
and $\exists ch. (ch \in (ins\ P) \wedge exprChannel\ ch\ E \wedge ch \notin (loc\ PQ))$
shows $ine\ PQ\ E$
 $\langle proof \rangle$

theorem $TBtheorem4b-P$:
assumes $ineM\ P\ M\ E$
and $subcomponents\ PQ = \{P, Q\}$
and $correctCompositionIn\ PQ$
and $\exists ch. ((ch \in (ins\ Q)) \wedge (exprChannel\ ch\ E) \wedge (ch \notin (loc\ PQ)) \wedge (ch \in M))$
shows $ineM\ PQ\ M\ E$
 $\langle proof \rangle$

theorem $TBtheorem4a-PQ$:
assumes $(ine\ P\ E) \vee (ine\ Q\ E)$
and $subcomponents\ PQ = \{P, Q\}$
and $correctCompositionIn\ PQ$
and $\exists ch. (((ch \in (ins\ P)) \vee (ch \in (ins\ Q))) \wedge (exprChannel\ ch\ E) \wedge (ch \notin (loc\ PQ)))$
shows $ine\ PQ\ E$
 $\langle proof \rangle$

theorem $TBtheorem4b-PQ$:
assumes $(ineM\ P\ M\ E) \vee (ineM\ Q\ M\ E)$

and *subcomponents* $PQ = \{P, Q\}$
and *correctCompositionIn* PQ
and $\exists ch. (((ch \in (ins P)) \vee (ch \in (ins Q))) \wedge$
 $(ch \in M) \wedge (exprChannel\ ch\ E) \wedge (ch \notin (loc\ PQ)))$
shows $ineM\ PQ\ M\ E$
 $\langle proof \rangle$

theorem *TBtheorem4a-notP1*:
assumes $ine\ P\ E$
and $\neg\ ine\ Q\ E$
and *subcomponents* $PQ = \{P, Q\}$
and *correctCompositionIn* PQ
and $\exists ch. ((ine-exprChannelSingle\ P\ ch\ E) \wedge (ch \in (loc\ PQ)))$
shows $\neg\ ine\ PQ\ E$
 $\langle proof \rangle$

theorem *TBtheorem4b-notP1*:
assumes $ineM\ P\ M\ E$
and $\neg\ ineM\ Q\ M\ E$
and *subcomponents* $PQ = \{P, Q\}$
and *correctCompositionIn* PQ
and $\exists ch. ((ine-exprChannelSingle\ P\ ch\ E) \wedge (ch \in M)$
 $\wedge (ch \in (loc\ PQ)))$
shows $\neg\ ineM\ PQ\ M\ E$
 $\langle proof \rangle$

theorem *TBtheorem4a-notP2*:
assumes $\neg\ ine\ Q\ E$
and *subcomponents* $PQ = \{P, Q\}$
and *correctCompositionIn* PQ
and *ine-exprChannelSet* $P\ ChSet\ E$
and $\forall (x :: chanID). ((x \in ChSet) \longrightarrow (x \in (loc\ PQ)))$
shows $\neg\ ine\ PQ\ E$
 $\langle proof \rangle$

theorem *TBtheorem4b-notP2*:
assumes $\neg\ ineM\ Q\ M\ E$
and *subcomponents* $PQ = \{P, Q\}$
and *correctCompositionIn* PQ
and *ine-exprChannelSet* $P\ ChSet\ E$
and $\forall (x :: chanID). ((x \in ChSet) \longrightarrow (x \in (loc\ PQ)))$
shows $\neg\ ineM\ PQ\ M\ E$
 $\langle proof \rangle$

theorem *TBtheorem4a-notPQ*:
assumes *subcomponents* $PQ = \{P, Q\}$
and *correctCompositionIn* PQ
and *ine-exprChannelSet* $P\ ChSetP\ E$
and *ine-exprChannelSet* $Q\ ChSetQ\ E$

and $\forall (x :: \text{chanID}). ((x \in \text{ChSet}P) \longrightarrow (x \in (\text{loc } PQ)))$
and $\forall (x :: \text{chanID}). ((x \in \text{ChSet}Q) \longrightarrow (x \in (\text{loc } PQ)))$
shows $\neg \text{ine } PQ E$
 $\langle \text{proof} \rangle$

lemma *ineM-Un1*:
assumes $\text{ineM } P A E$
shows $\text{ineM } P (A \text{ Un } B) E$
 $\langle \text{proof} \rangle$

theorem *TBtheorem4b-notPQ*:
assumes $\text{subcomponents } PQ = \{P, Q\}$
and $\text{correctCompositionIn } PQ$
and $\text{ine-exprChannelSet } P \text{ ChSet}P E$
and $\text{ine-exprChannelSet } Q \text{ ChSet}Q E$
and $\forall (x :: \text{chanID}). ((x \in \text{ChSet}P) \longrightarrow (x \in (\text{loc } PQ)))$
and $\forall (x :: \text{chanID}). ((x \in \text{ChSet}Q) \longrightarrow (x \in (\text{loc } PQ)))$
shows $\neg \text{ineM } PQ M E$
 $\langle \text{proof} \rangle$

lemma *ine-nonempty-exprChannelSet*:
assumes $\text{ine-exprChannelSet } P \text{ ChSet } E$
and $\text{ChSet} \neq \{\}$
shows $\text{ine } P E$
 $\langle \text{proof} \rangle$

lemma *ine-empty-exprChannelSet*:
assumes $\text{ine-exprChannelSet } P \text{ ChSet } E$
and $\text{ChSet} = \{\}$
shows $\neg \text{ine } P E$
 $\langle \text{proof} \rangle$

theorem *TBtheorem5a-empty*:
assumes $(\text{eout } P E) \vee (\text{eout } Q E)$
and $\text{subcomponents } PQ = \{P, Q\}$
and $\text{correctCompositionOut } PQ$
and $\text{loc } PQ = \{\}$
shows $\text{eout } PQ E$
 $\langle \text{proof} \rangle$

theorem *TBtheorem45a-P*:
assumes $\text{eout } P E$
and $\text{subcomponents } PQ = \{P, Q\}$
and $\text{correctCompositionOut } PQ$
and $\exists \text{ch}. ((\text{ch} \in (\text{out } P)) \wedge (\text{exprChannel } \text{ch } E) \wedge$
 $\quad (\text{ch} \notin (\text{loc } PQ)))$
shows $\text{eout } PQ E$
 $\langle \text{proof} \rangle$

theorem *TBtheore54b-P:*

assumes *eoutM P M E*

and *subcomponents PQ = {P,Q}*

and *correctCompositionOut PQ*

and $\exists ch. ((ch \in (out\ Q)) \wedge (exprChannel\ ch\ E) \wedge$
 $(ch \notin (loc\ PQ)) \wedge (ch \in M))$

shows *eoutM PQ M E*

<proof>

theorem *TBtheorem5a-PQ:*

assumes $(eout\ P\ E) \vee (eout\ Q\ E)$

and *subcomponents PQ = {P,Q}*

and *correctCompositionOut PQ*

and $\exists ch. (((ch \in (out\ P)) \vee (ch \in (out\ Q))) \wedge$
 $(exprChannel\ ch\ E) \wedge (ch \notin (loc\ PQ)))$

shows *eout PQ E*

<proof>

theorem *TBtheorem5b-PQ:*

assumes $(eoutM\ P\ M\ E) \vee (eoutM\ Q\ M\ E)$

and *subcomponents PQ = {P,Q}*

and *correctCompositionOut PQ*

and $\exists ch. (((ch \in (out\ P)) \vee (ch \in (out\ Q))) \wedge (ch \in M)$
 $\wedge (exprChannel\ ch\ E) \wedge (ch \notin (loc\ PQ)))$

shows *eoutM PQ M E*

<proof>

theorem *TBtheorem5a-notP1:*

assumes *eout P E*

and $\neg eout\ Q\ E$

and *subcomponents PQ = {P,Q}*

and *correctCompositionOut PQ*

and $\exists ch. ((out-exprChannelSingle\ P\ ch\ E) \wedge (ch \in (loc\ PQ)))$

shows $\neg eout\ PQ\ E$

<proof>

theorem *TBtheorem5b-notP1:*

assumes *eoutM P M E*

and $\neg eoutM\ Q\ M\ E$

and *subcomponents PQ = {P,Q}*

and *correctCompositionOut PQ*

and $\exists ch. ((out-exprChannelSingle\ P\ ch\ E) \wedge (ch \in M)$
 $\wedge (ch \in (loc\ PQ)))$

shows $\neg eoutM\ PQ\ M\ E$

<proof>

theorem *TBtheorem5a-notP2:*

assumes $\neg eout\ Q\ E$

and *subcomponents PQ = {P,Q}*

and *correctCompositionOut* PQ
and *out-exprChannelSet* P $ChSet$ E
and $\forall (x :: chanID). ((x \in ChSet) \longrightarrow (x \in (loc PQ)))$
shows $\neg eout PQ E$
 $\langle proof \rangle$

theorem *TBtheorem5b-notP2*:
assumes $\neg eoutM Q M E$
and *subcomponents* $PQ = \{P, Q\}$
and *correctCompositionOut* PQ
and *out-exprChannelSet* P $ChSet$ E
and $\forall (x :: chanID). ((x \in ChSet) \longrightarrow (x \in (loc PQ)))$
shows $\neg eoutM PQ M E$
 $\langle proof \rangle$

theorem *TBtheorem5a-notPQ*:
assumes *subcomponents* $PQ = \{P, Q\}$
and *correctCompositionOut* PQ
and *out-exprChannelSet* P $ChSetP$ E
and *out-exprChannelSet* Q $ChSetQ$ E
and $\forall (x :: chanID). ((x \in ChSetP) \longrightarrow (x \in (loc PQ)))$
and $\forall (x :: chanID). ((x \in ChSetQ) \longrightarrow (x \in (loc PQ)))$
shows $\neg eout PQ E$
 $\langle proof \rangle$

theorem *TBtheorem5b-notPQ*:
assumes *subcomponents* $PQ = \{P, Q\}$
and *correctCompositionOut* PQ
and *out-exprChannelSet* P $ChSetP$ E
and *out-exprChannelSet* Q $ChSetQ$ E
and $M = ChSetP \cup ChSetQ$
and $\forall (x :: chanID). ((x \in ChSetP) \longrightarrow (x \in (loc PQ)))$
and $\forall (x :: chanID). ((x \in ChSetQ) \longrightarrow (x \in (loc PQ)))$
shows $\neg eoutM PQ M E$
 $\langle proof \rangle$

end

4 Local Secrets of a component

theory *CompLocalSecrets*
imports *Secrecy*
begin

- Set of local secrets: the set of secrets which does not belong to
- the set of private keys and unguessable values, but are transmitted
- via local channels or belongs to the local secrets of its subcomponents

axiomatization

LocalSecrets $:: specID \Rightarrow KS$ set

where

LocalSecretsDef:

LocalSecrets A =

$\{(m :: KS). m \notin \text{specKeysSecrets } A \wedge$
 $((\exists x y. ((x \in \text{loc } A) \wedge m = (kKS y) \wedge (\text{exprChannel } x (kE y))))$
 $\mid (\exists x z. ((x \in \text{loc } A) \wedge m = (sKS z) \wedge (\text{exprChannel } x (sE z)))))\}$
 $\cup (\cup (\text{LocalSecrets } ` (\text{subcomponents } A)))$

lemma *LocalSecretsComposition1*:

assumes $ls \in \text{LocalSecrets } P$

and $\text{subcomponents } PQ = \{P, Q\}$

shows $ls \in \text{LocalSecrets } PQ$

<proof>

lemma *LocalSecretsComposition-exprChannel-k*:

assumes $\text{exprChannel } x (kE \text{ Keys})$

and $\neg \text{ine } P (kE \text{ Keys})$

and $\neg \text{ine } Q (kE \text{ Keys})$

and $\neg (x \notin \text{ins } P \wedge x \notin \text{ins } Q)$

shows *False*

<proof>

lemma *LocalSecretsComposition-exprChannel-s*:

assumes $\text{exprChannel } x (sE \text{ Secrets})$

and $\neg \text{ine } P (sE \text{ Secrets})$

and $\neg \text{ine } Q (sE \text{ Secrets})$

and $\neg (x \notin \text{ins } P \wedge x \notin \text{ins } Q)$

shows *False*

<proof>

lemma *LocalSecretsComposition-neg1-k*:

assumes $\text{subcomponents } PQ = \{P, Q\}$

and $\text{correctCompositionLoc } PQ$

and $\neg \text{ine } P (kE \text{ Keys})$

and $\neg \text{ine } Q (kE \text{ Keys})$

and $kKS \text{ Keys} \notin \text{LocalSecrets } P$

and $kKS \text{ Keys} \notin \text{LocalSecrets } Q$

shows $kKS \text{ Keys} \notin \text{LocalSecrets } PQ$

<proof>

lemma *LocalSecretsComposition-neg-k*:

assumes $\text{subcomponents } PQ = \{P, Q\}$

and $\text{correctCompositionLoc } PQ$

and $\text{correctCompositionKS } PQ$

and $(kKS m) \notin \text{specKeysSecrets } P$

and $(kKS m) \notin \text{specKeysSecrets } Q$

and $\neg \text{ine } P (kE m)$

and $\neg \text{ine } Q (kE m)$

and $(kKS m) \notin ((\text{LocalSecrets } P) \cup (\text{LocalSecrets } Q))$

shows $(kKS\ m) \notin (LocalSecrets\ PQ)$
 ⟨proof⟩

lemma *LocalSecretsComposition-neg-s:*
assumes $subPQ:subcomponents\ PQ = \{P, Q\}$
and $cCompLoc:correctCompositionLoc\ PQ$
and $cCompKS:correctCompositionKS\ PQ$
and $notKSP:(sKS\ m) \notin specKeysSecrets\ P$
and $notKSQ:(sKS\ m) \notin specKeysSecrets\ Q$
and $\neg\ ine\ P\ (sE\ m)$
and $\neg\ ine\ Q\ (sE\ m)$
and $notLocSeqPQ:(sKS\ m) \notin ((LocalSecrets\ P) \cup (LocalSecrets\ Q))$
shows $(sKS\ m) \notin (LocalSecrets\ PQ)$
 ⟨proof⟩

lemma *LocalSecretsComposition-neg:*
assumes $subcomponents\ PQ = \{P, Q\}$
and $correctCompositionLoc\ PQ$
and $correctCompositionKS\ PQ$
and $ks \notin specKeysSecrets\ P$
and $ks \notin specKeysSecrets\ Q$
and $h1:\forall\ m. ks = kKS\ m \longrightarrow (\neg\ ine\ P\ (kE\ m) \wedge \neg\ ine\ Q\ (kE\ m))$
and $h2:\forall\ m. ks = sKS\ m \longrightarrow (\neg\ ine\ P\ (sE\ m) \wedge \neg\ ine\ Q\ (sE\ m))$
and $ks \notin ((LocalSecrets\ P) \cup (LocalSecrets\ Q))$
shows $ks \notin (LocalSecrets\ PQ)$
 ⟨proof⟩

lemma *LocalSecretsComposition-neg1-s:*
assumes $subcomponents\ PQ = \{P, Q\}$
and $correctCompositionLoc\ PQ$
and $\neg\ ine\ P\ (sE\ s)$
and $\neg\ ine\ Q\ (sE\ s)$
and $sKS\ s \notin LocalSecrets\ P$
and $sKS\ s \notin LocalSecrets\ Q$
shows $sKS\ s \notin LocalSecrets\ PQ$
 ⟨proof⟩

lemma *LocalSecretsComposition-neg1:*
assumes $subcomponents\ PQ = \{P, Q\}$
and $correctCompositionLoc\ PQ$
and $h1:\forall\ m. ks = kKS\ m \longrightarrow (\neg\ ine\ P\ (kE\ m) \wedge \neg\ ine\ Q\ (kE\ m))$
and $h2:\forall\ m. ks = sKS\ m \longrightarrow (\neg\ ine\ P\ (sE\ m) \wedge \neg\ ine\ Q\ (sE\ m))$
and $ks \notin LocalSecrets\ P$
and $ks \notin LocalSecrets\ Q$
shows $ks \notin LocalSecrets\ PQ$
 ⟨proof⟩

lemma *LocalSecretsComposition-ine1-k:*
assumes $kKS\ k \in LocalSecrets\ PQ$

and *subcomponents* $PQ = \{P, Q\}$
and *correctCompositionLoc* PQ
and $\neg \text{ine } Q (kE k)$
and $kKS k \notin \text{LocalSecrets } P$
and $kKS k \notin \text{LocalSecrets } Q$
shows $\text{ine } P (kE k)$
<proof>

lemma *LocalSecretsComposition-ine1-s:*
assumes $sKS s \in \text{LocalSecrets } PQ$
and *subcomponents* $PQ = \{P, Q\}$
and *correctCompositionLoc* PQ
and $\neg \text{ine } Q (sE s)$
and $sKS s \notin \text{LocalSecrets } P$
and $sKS s \notin \text{LocalSecrets } Q$
shows $\text{ine } P (sE s)$
<proof>

lemma *LocalSecretsComposition-ine2-k:*
assumes $kKS k \in \text{LocalSecrets } PQ$
and *subcomponents* $PQ = \{P, Q\}$
and *correctCompositionLoc* PQ
and $\neg \text{ine } P (kE k)$
and $kKS k \notin \text{LocalSecrets } P$
and $kKS k \notin \text{LocalSecrets } Q$
shows $\text{ine } Q (kE k)$
<proof>

lemma *LocalSecretsComposition-ine2-s:*
assumes $sKS s \in \text{LocalSecrets } PQ$
and *subcomponents* $PQ = \{P, Q\}$
and *correctCompositionLoc* PQ
and $\neg \text{ine } P (sE s)$
and $sKS s \notin \text{LocalSecrets } P$
and $sKS s \notin \text{LocalSecrets } Q$
shows $\text{ine } Q (sE s)$
<proof>

lemma *LocalSecretsComposition-neg-loc-k:*
assumes $kKS key \notin \text{LocalSecrets } P$
and *exprChannel* $ch (kE key)$
and $kKS key \notin \text{specKeysSecrets } P$
shows $ch \notin \text{loc } P$
<proof>

lemma *LocalSecretsComposition-neg-loc-s:*
assumes $sKS secret \notin \text{LocalSecrets } P$
and *exprChannel* $ch (sE secret)$
and $sKS secret \notin \text{specKeysSecrets } P$

shows $ch \notin \text{loc } P$
(*proof*)

lemma *correctCompositionKS-exprChannel-k-P:*

assumes $\text{subcomponents } PQ = \{P, Q\}$
and *correctCompositionKS* PQ
and $kKS \text{ key} \notin \text{LocalSecrets } PQ$
and $ch \in \text{ins } P$
and *exprChannel* $ch \text{ (kE key)}$
and $kKS \text{ key} \notin \text{specKeysSecrets } PQ$
and *correctCompositionIn* PQ
shows $ch \in \text{ins } PQ \wedge \text{exprChannel } ch \text{ (kE key)}$
(*proof*)

lemma *correctCompositionKS-exprChannel-k-Pex:*

assumes $\text{subcomponents } PQ = \{P, Q\}$
and *correctCompositionKS* PQ
and $kKS \text{ key} \notin \text{LocalSecrets } PQ$
and $ch \in \text{ins } P$
and *exprChannel* $ch \text{ (kE key)}$
and $kKS \text{ key} \notin \text{specKeysSecrets } PQ$
and *correctCompositionIn* PQ
shows $\exists ch. ch \in \text{ins } PQ \wedge \text{exprChannel } ch \text{ (kE key)}$
(*proof*)

lemma *correctCompositionKS-exprChannel-k-Q:*

assumes $\text{subcomponents } PQ = \{P, Q\}$
and *correctCompositionKS* PQ
and $kKS \text{ key} \notin \text{LocalSecrets } PQ$
and $ch \in \text{ins } Q$
and $h1: \text{exprChannel } ch \text{ (kE key)}$
and $kKS \text{ key} \notin \text{specKeysSecrets } PQ$
and *correctCompositionIn* PQ
shows $ch \in \text{ins } PQ \wedge \text{exprChannel } ch \text{ (kE key)}$
(*proof*)

lemma *correctCompositionKS-exprChannel-k-Qex:*

assumes $\text{subcomponents } PQ = \{P, Q\}$
and *correctCompositionKS* PQ
and $kKS \text{ key} \notin \text{LocalSecrets } PQ$
and $ch \in \text{ins } Q$
and *exprChannel* $ch \text{ (kE key)}$
and $kKS \text{ key} \notin \text{specKeysSecrets } PQ$
and *correctCompositionIn* PQ
shows $\exists ch. ch \in \text{ins } PQ \wedge \text{exprChannel } ch \text{ (kE key)}$
(*proof*)

lemma *correctCompositionKS-exprChannel-s-P:*

assumes $\text{subcomponents } PQ = \{P, Q\}$

```

    and correctCompositionKS PQ
    and sKS secret  $\notin$  LocalSecrets PQ
    and ch  $\in$  ins P
    and exprChannel ch (sE secret)
    and sKS secret  $\notin$  specKeysSecrets PQ
    and correctCompositionIn PQ
shows   ch  $\in$  ins PQ  $\wedge$  exprChannel ch (sE secret)
<proof>

lemma correctCompositionKS-exprChannel-s-Pex:
assumes subcomponents PQ = {P,Q}
    and correctCompositionKS PQ
    and sKS secret  $\notin$  LocalSecrets PQ
    and ch  $\in$  ins P
    and exprChannel ch (sE secret)
    and sKS secret  $\notin$  specKeysSecrets PQ
    and correctCompositionIn PQ
shows    $\exists$  ch. ch  $\in$  ins PQ  $\wedge$  exprChannel ch (sE secret)
<proof>

lemma correctCompositionKS-exprChannel-s-Q:
assumes subcomponents PQ = {P,Q}
    and correctCompositionKS PQ
    and sKS secret  $\notin$  LocalSecrets PQ
    and ch  $\in$  ins Q
    and h1:exprChannel ch (sE secret)
    and sKS secret  $\notin$  specKeysSecrets PQ
    and correctCompositionIn PQ
shows   ch  $\in$  ins PQ  $\wedge$  exprChannel ch (sE secret)
<proof>

lemma correctCompositionKS-exprChannel-s-Qex:
assumes subcomponents PQ = {P,Q}
    and correctCompositionKS PQ
    and sKS secret  $\notin$  LocalSecrets PQ
    and ch  $\in$  ins Q
    and exprChannel ch (sE secret)
    and sKS secret  $\notin$  specKeysSecrets PQ
    and correctCompositionIn PQ
shows    $\exists$  ch. ch  $\in$  ins PQ  $\wedge$  exprChannel ch (sE secret)
<proof>

end

```

5 Knowledge of Keys and Secrets

```

theory KnowledgeKeysSecrets
imports CompLocalSecrets
begin

```

An component A knows a secret m (or some secret expression m) that does not belong to its local secrets, if

- A may eventually get the secret m ,
- m belongs to the set LS_A of its local secrets,
- A knows some list of expressions m_2 which is an concatenations of m and some list of expressions m_1 ,
- m is a concatenation of some lists of secrets m_1 and m_2 , and A knows both these secrets,
- A knows some secret key k^{-1} and the result of the encryption of the m with the corresponding public key,
- A knows some public key k and the result of the signature creation of the m with the corresponding private key,
- m is an encryption of some secret m_1 with a public key k , and A knows both m_1 and k ,
- m is the result of the signature creation of the m_1 with the key k , and A knows both m_1 and k .

primrec

$know :: specID \Rightarrow KS \Rightarrow bool$

where

$know\ A\ (kKS\ m) =$
 $((ine\ A\ (kE\ m)) \vee ((kKS\ m) \in (LocalSecrets\ A))) \mid$
 $know\ A\ (sKS\ m) =$
 $((ine\ A\ (sE\ m)) \vee ((sKS\ m) \in (LocalSecrets\ A)))$

axiomatization

$knows :: specID \Rightarrow Expression\ list \Rightarrow bool$

where

$knows\ emptyexpression:$

$knows\ C\ [] = True$ **and**

$know1k:$

$knows\ C\ [KS2Expression\ (kKS\ m1)] = know\ C\ (kKS\ m1)$ **and**

$know1s:$

$knows\ C\ [KS2Expression\ (sKS\ m2)] = know\ C\ (sKS\ m2)$ **and**

$knows2a:$

$knows\ A\ (e1\ @\ e) \longrightarrow knows\ A\ e$ **and**

$knows2b:$

$knows\ A\ (e\ @\ e1) \longrightarrow knows\ A\ e$ **and**

$knows3:$

$(knows\ A\ e1) \wedge (knows\ A\ e2) \longrightarrow knows\ A\ (e1\ @\ e2)$ **and**

$knows4:$

$(IncrDecrKeys\ k1\ k2) \wedge (know\ A\ (kKS\ k2)) \wedge (knows\ A\ (Enc\ k1\ e))$
 $\longrightarrow knows\ A\ e$

and

$knows5:$

$(IncrDecrKeys\ k1\ k2) \wedge (know\ A\ (kKS\ k1)) \wedge (knows\ A\ (Sign\ k2\ e))$

$\longrightarrow \text{knows } A \ e$
and
knows6:
 $(\text{know } A \ (kKS \ k)) \wedge (\text{knows } A \ e1) \longrightarrow \text{knows } A \ (\text{Enc } k \ e1)$
and
knows7:
 $(\text{know } A \ (kKS \ k)) \wedge (\text{knows } A \ e1) \longrightarrow \text{knows } A \ (\text{Sign } k \ e1)$

primrec *eoutKnowCorrect* :: *specID* \Rightarrow *KS* \Rightarrow *bool*
where
eout-know-k:
 $\text{eoutKnowCorrect } C \ (kKS \ m) =$
 $((\text{eout } C \ (kE \ m)) \longleftrightarrow (m \in (\text{specKeys } C) \vee (\text{know } C \ (kKS \ m)))) \mid$
eout-know-s:
 $\text{eoutKnowCorrect } C \ (sKS \ m) =$
 $((\text{eout } C \ (sE \ m)) \longleftrightarrow (m \in (\text{specSecrets } C) \vee (\text{know } C \ (sKS \ m))))$

definition *eoutKnowsECorrect* :: *specID* \Rightarrow *Expression* \Rightarrow *bool*
where
 $\text{eoutKnowsECorrect } C \ e \equiv$
 $((\text{eout } C \ e) \longleftrightarrow$
 $(\exists k. e = (kE \ k) \wedge (k \in \text{specKeys } C)) \vee$
 $(\exists s. e = (sE \ s) \wedge (s \in \text{specSecrets } C)) \vee$
 $(\text{knows } C \ [e])))$

lemma *eoutKnowCorrect-L1k*:
assumes *eoutKnowCorrect* *C* (*kKS* *m*)
and *eout* *C* (*kE* *m*)
shows $m \in (\text{specKeys } C) \vee (\text{know } C \ (kKS \ m))$
 $\langle \text{proof} \rangle$

lemma *eoutKnowCorrect-L1s*:
assumes *eoutKnowCorrect* *C* (*sKS* *m*)
and *eout* *C* (*sE* *m*)
shows $m \in (\text{specSecrets } C) \vee (\text{know } C \ (sKS \ m))$
 $\langle \text{proof} \rangle$

lemma *eoutKnowsECorrect-L1*:
assumes *eoutKnowsECorrect* *C* *e*
and *eout* *C* *e*
shows $(\exists k. e = (kE \ k) \wedge (k \in \text{specKeys } C)) \vee$
 $(\exists s. e = (sE \ s) \wedge (s \in \text{specSecrets } C)) \vee$
 $(\text{knows } C \ [e])$
 $\langle \text{proof} \rangle$

lemma *know2knows-k*:
assumes *know* *A* (*kKS* *m*)
shows *knows* *A* [*kE* *m*]
 $\langle \text{proof} \rangle$

lemma *knows2know-k*:
assumes *knows A [kE m]*
shows *know A (kKS m)*
 \langle *proof* \rangle

lemma *know2knowsPQ-k*:
assumes *know P (kKS m) \vee know Q (kKS m)*
shows *knows P [kE m] \vee knows Q [kE m]*
 \langle *proof* \rangle

lemma *knows2knowPQ-k*:
assumes *knows P [kE m] \vee knows Q [kE m]*
shows *know P (kKS m) \vee know Q (kKS m)*
 \langle *proof* \rangle

lemma *knows1k*:
know A (kKS m) = knows A [kE m]
 \langle *proof* \rangle

lemma *know2knows-neg-k*:
assumes \neg *know A (kKS m)*
shows \neg *knows A [kE m]*
 \langle *proof* \rangle

lemma *knows2know-neg-k*:
assumes \neg *knows A [kE m]*
shows \neg *know A (kKS m)*
 \langle *proof* \rangle

lemma *know2knows-s*:
assumes *know A (sKS m)*
shows *knows A [sE m]*
 \langle *proof* \rangle

lemma *knows2know-s*:
assumes *knows A [sE m]*
shows *know A (sKS m)*
 \langle *proof* \rangle

lemma *know2knowsPQ-s*:
assumes *know P (sKS m) \vee know Q (sKS m)*
shows *knows P [sE m] \vee knows Q [sE m]*
 \langle *proof* \rangle

lemma *knows2knowPQ-s*:
assumes *knows P [sE m] \vee knows Q [sE m]*
shows *know P (sKS m) \vee know Q (sKS m)*
 \langle *proof* \rangle

lemma *knows1s*:
 $know\ A\ (sKS\ m) = knows\ A\ [sE\ m]$
 ⟨proof⟩

lemma *know2knows-neg-s*:
assumes $\neg\ know\ A\ (sKS\ m)$
shows $\neg\ knows\ A\ [sE\ m]$
 ⟨proof⟩

lemma *knows2know-neg-s*:
assumes $\neg\ knows\ A\ [sE\ m]$
shows $\neg\ know\ A\ (sKS\ m)$
 ⟨proof⟩

lemma *knows2*:
assumes $e2 = e1\ @\ e \vee e2 = e\ @\ e1$
and $knows\ A\ e2$
shows $knows\ A\ e$
 ⟨proof⟩

lemma *correctCompositionInLoc-exprChannel*:
assumes $subcomponents\ PQ = \{P, Q\}$
and *correctCompositionIn* PQ
and $ch : ins\ P$
and *exprChannel* $ch\ m$
and $\forall\ x. x \in ins\ PQ \longrightarrow \neg\ exprChannel\ x\ m$
shows $ch : loc\ PQ$
 ⟨proof⟩

lemma *eout-know-nonKS-k*:
assumes $m \notin specKeys\ A$
and $eout\ A\ (kE\ m)$
and *eoutKnowCorrect* $A\ (kKS\ m)$
shows $know\ A\ (kKS\ m)$
 ⟨proof⟩

lemma *eout-know-nonKS-s*:
assumes $m \notin specSecrets\ A$
and $eout\ A\ (sE\ m)$
and *eoutKnowCorrect* $A\ (sKS\ m)$
shows $know\ A\ (sKS\ m)$
 ⟨proof⟩

lemma *not-know-k-not-ine*:
assumes $\neg\ know\ A\ (kKS\ m)$
shows $\neg\ ine\ A\ (kE\ m)$
 ⟨proof⟩

lemma *not-know-s-not-ine*:
assumes $\neg \text{know } A \text{ (sKS } m)$
shows $\neg \text{ine } A \text{ (sE } m)$
 $\langle \text{proof} \rangle$

lemma *not-know-k-not-eout*:
assumes $m \notin \text{specKeys } A$
and $\neg \text{know } A \text{ (kKS } m)$
and $\text{eoutKnowCorrect } A \text{ (kKS } m)$
shows $\neg \text{eout } A \text{ (kE } m)$
 $\langle \text{proof} \rangle$

lemma *not-know-s-not-eout*:
assumes $m \notin \text{specSecrets } A$
and $\neg \text{know } A \text{ (sKS } m)$
and $\text{eoutKnowCorrect } A \text{ (sKS } m)$
shows $\neg \text{eout } A \text{ (sE } m)$
 $\langle \text{proof} \rangle$

lemma *adv-not-know1*:
assumes $\text{out } P \subseteq \text{ins } A$
and $\neg \text{know } A \text{ (kKS } m)$
shows $\neg \text{eout } P \text{ (kE } m)$
 $\langle \text{proof} \rangle$

lemma *adv-not-know2*:
assumes $\text{out } P \subseteq \text{ins } A$
and $\neg \text{know } A \text{ (sKS } m)$
shows $\neg \text{eout } P \text{ (sE } m)$
 $\langle \text{proof} \rangle$

lemma *LocalSecrets-L1*:
assumes $(kKS) \text{ key} \in \text{LocalSecrets } P$
and $(kKS \text{ key}) \notin \bigcup (\text{LocalSecrets ' subcomponents } P)$
shows $kKS \text{ key} \notin \text{specKeysSecrets } P$
 $\langle \text{proof} \rangle$

lemma *LocalSecrets-L2*:
assumes $kKS \text{ key} \in \text{LocalSecrets } P$
and $kKS \text{ key} \in \text{specKeysSecrets } P$
shows $kKS \text{ key} \in \bigcup (\text{LocalSecrets ' subcomponents } P)$
 $\langle \text{proof} \rangle$

lemma *know-composition1*:
assumes $\text{notKSP}:m \notin \text{specKeysSecrets } P$
and $\text{notKSQ}:m \notin \text{specKeysSecrets } Q$
and $\text{know } P \text{ } m$
and $\text{subPQ}:\text{subcomponents } PQ = \{P, Q\}$
and $\text{cCompI}:\text{correctCompositionIn } PQ$

and *cCompKS:correctCompositionKS PQ*
shows *know PQ m*
 ⟨*proof*⟩

lemma *know-composition2:*
assumes $m \notin \text{specKeysSecrets } P$
and $m \notin \text{specKeysSecrets } Q$
and *know Q m*
and $\text{subcomponents } PQ = \{P, Q\}$
and *correctCompositionIn PQ*
and *correctCompositionKS PQ*
shows *know PQ m*
 ⟨*proof*⟩

lemma *know-composition:*
assumes $m \notin \text{specKeysSecrets } P$
and $m \notin \text{specKeysSecrets } Q$
and $\text{know } P \ m \ \vee \ \text{know } Q \ m$
and $\text{subcomponents } PQ = \{P, Q\}$
and *correctCompositionIn PQ*
and *correctCompositionKS PQ*
shows *know PQ m*
 ⟨*proof*⟩

theorem *know-composition-neg-ine-k:*
assumes $\neg \text{know } P \ (kKS \ \text{key})$
and $\neg \text{know } Q \ (kKS \ \text{key})$
and $\text{subcomponents } PQ = \{P, Q\}$
and *correctCompositionIn PQ*
shows $\neg (\text{ine } PQ \ (kE \ \text{key}))$
 ⟨*proof*⟩

theorem *know-composition-neg-ine-s:*
assumes $\neg \text{know } P \ (sKS \ \text{secret})$
and $\neg \text{know } Q \ (sKS \ \text{secret})$
and $\text{subcomponents } PQ = \{P, Q\}$
and *correctCompositionIn PQ*
shows $\neg (\text{ine } PQ \ (sE \ \text{secret}))$
 ⟨*proof*⟩

lemma *know-composition-neg1:*
assumes $\text{notknowP} : \neg \text{know } P \ m$
and $\text{notknowQ} : \neg \text{know } Q \ m$
and $\text{subPQ} : \text{subcomponents } PQ = \{P, Q\}$
and *cCompLoc:correctCompositionLoc PQ*
and *cCompI:correctCompositionIn PQ*
shows $\neg \text{know } PQ \ m$
 ⟨*proof*⟩

lemma *know-decomposition*:
assumes *knowPQ:know PQ m*
and *subPQ:subcomponents PQ = {P,Q}*
and *cCompI:correctCompositionIn PQ*
and *cCompLoc:correctCompositionLoc PQ*
shows *know P m \vee know Q m*
 \langle *proof* \rangle

lemma *eout-knows-nonKS-k*:
assumes *m \notin (specKeys A)*
and *eout A (kE m)*
and *eoutKnowsECorrect A (kE m)*
shows *knows A [kE m]*
 \langle *proof* \rangle

lemma *eout-knows-nonKS-s*:
assumes *h1:m \notin specSecrets A*
and *h2:eout A (sE m)*
and *h3:eoutKnowsECorrect A (sE m)*
shows *knows A [sE m]*
 \langle *proof* \rangle

lemma *not-knows-k-not-ine*:
assumes \neg *knows A [kE m]*
shows \neg *ine A (kE m)*
 \langle *proof* \rangle

lemma *not-knows-s-not-ine*:
assumes \neg *knows A [sE m]*
shows \neg *ine A (sE m)*
 \langle *proof* \rangle

lemma *not-knows-k-not-eout*:
assumes *m \notin specKeys A*
and \neg *knows A [kE m]*
and *eoutKnowsECorrect A (kE m)*
shows \neg *eout A (kE m)*
 \langle *proof* \rangle

lemma *not-knows-s-not-eout*:
assumes *m \notin specSecrets A*
and \neg *knows A [sE m]*
and *eoutKnowsECorrect A (sE m)*
shows \neg *eout A (sE m)*
 \langle *proof* \rangle

lemma *adv-not-knows1*:
assumes *out P \subseteq ins A*
and \neg *knows A [kE m]*

shows $\neg \text{eout } P (kE m)$
<proof>

lemma *adv-not-knows2*:
assumes $\text{out } P \subseteq \text{ins } A$
 and $\neg \text{knows } A [sE m]$
shows $\neg \text{eout } P (sE m)$
<proof>

lemma *knows-decomposition-1-k*:
assumes $kKS a \notin \text{specKeysSecrets } P$
 and $kKS a \notin \text{specKeysSecrets } Q$
 and $\text{subcomponents } PQ = \{P, Q\}$
 and $\text{knows } PQ [kE a]$
 and $\text{correctCompositionIn } PQ$
 and $\text{correctCompositionLoc } PQ$
shows $\text{knows } P [kE a] \vee \text{knows } Q [kE a]$
<proof>

lemma *knows-decomposition-1-s*:
assumes $sKS a \notin \text{specKeysSecrets } P$
 and $sKS a \notin \text{specKeysSecrets } Q$
 and $\text{subcomponents } PQ = \{P, Q\}$
 and $\text{knows } PQ [sE a]$
 and $\text{correctCompositionIn } PQ$
 and $\text{correctCompositionLoc } PQ$
shows $\text{knows } P [sE a] \vee \text{knows } Q [sE a]$
<proof>

lemma *knows-decomposition-1*:
assumes $\text{subcomponents } PQ = \{P, Q\}$
 and $\text{knows } PQ [a]$
 and $\text{correctCompositionIn } PQ$
 and $\text{correctCompositionLoc } PQ$
 and $(\exists z. a = kE z) \vee (\exists z. a = sE z)$
 and $\forall z. a = kE z \longrightarrow$
 $kKS z \notin \text{specKeysSecrets } P \wedge kKS z \notin \text{specKeysSecrets } Q$
 and $\forall z. a = sE z \longrightarrow$
 $sKS z \notin \text{specKeysSecrets } P \wedge sKS z \notin \text{specKeysSecrets } Q$
shows $\text{knows } P [a] \vee \text{knows } Q [a]$
<proof>

lemma *knows-composition1-k*:
assumes $(kKS m) \notin \text{specKeysSecrets } P$
 and $(kKS m) \notin \text{specKeysSecrets } Q$
 and $\text{knows } P [kE m]$
 and $\text{subcomponents } PQ = \{P, Q\}$
 and $\text{correctCompositionIn } PQ$
 and $\text{correctCompositionKS } PQ$

shows $\textit{knows PQ [kE m]}$
 $\langle \textit{proof} \rangle$

lemma $\textit{knows-composition1-s}$:
assumes $(sKS m) \notin \textit{specKeysSecrets P}$
 and $(sKS m) \notin \textit{specKeysSecrets Q}$
 and $\textit{knows P [sE m]}$
 and $\textit{subcomponents PQ} = \{P, Q\}$
 and $\textit{correctCompositionIn PQ}$
 and $\textit{correctCompositionKS PQ}$
shows $\textit{knows PQ [sE m]}$
 $\langle \textit{proof} \rangle$

lemma $\textit{knows-composition2-k}$:
assumes $(kKS m) \notin \textit{specKeysSecrets P}$
 and $(kKS m) \notin \textit{specKeysSecrets Q}$
 and $\textit{knows Q [kE m]}$
 and $\textit{subcomponents PQ} = \{P, Q\}$
 and $\textit{correctCompositionIn PQ}$
 and $\textit{correctCompositionKS PQ}$
shows $\textit{knows PQ [kE m]}$
 $\langle \textit{proof} \rangle$

lemma $\textit{knows-composition2-s}$:
assumes $(sKS m) \notin \textit{specKeysSecrets P}$
 and $(sKS m) \notin \textit{specKeysSecrets Q}$
 and $\textit{knows Q [sE m]}$
 and $\textit{subcomponents PQ} = \{P, Q\}$
 and $\textit{correctCompositionIn PQ}$
 and $\textit{correctCompositionKS PQ}$
shows $\textit{knows PQ [sE m]}$
 $\langle \textit{proof} \rangle$

lemma $\textit{knows-composition-neg1-k}$:
assumes $kKS m \notin \textit{specKeysSecrets P}$
 and $kKS m \notin \textit{specKeysSecrets Q}$
 and $\neg \textit{knows P [kE m]}$
 and $\neg \textit{knows Q [kE m]}$
 and $\textit{subcomponents PQ} = \{P, Q\}$
 and $\textit{correctCompositionLoc PQ}$
 and $\textit{correctCompositionIn PQ}$
 and $\textit{correctCompositionKS PQ}$
shows $\neg \textit{knows PQ [kE m]}$
 $\langle \textit{proof} \rangle$

lemma $\textit{knows-composition-neg1-s}$:
assumes $sKS m \notin \textit{specKeysSecrets P}$
 and $sKS m \notin \textit{specKeysSecrets Q}$
 and $\neg \textit{knows P [sE m]}$

and $\neg \text{knows } Q [sE m]$
and $\text{subcomponents } PQ = \{P, Q\}$
and $\text{correctCompositionLoc } PQ$
and $\text{correctCompositionIn } PQ$
and $\text{correctCompositionKS } PQ$
shows $\neg \text{knows } PQ [sE m]$
 $\langle \text{proof} \rangle$

lemma *knows-concat-1*:
assumes $\text{knows } P (a \# e)$
shows $\text{knows } P [a]$
 $\langle \text{proof} \rangle$

lemma *knows-concat-2*:
assumes $\text{knows } P (a \# e)$
shows $\text{knows } P e$
 $\langle \text{proof} \rangle$

lemma *knows-concat-3*:
assumes $\text{knows } P [a]$
and $\text{knows } P e$
shows $\text{knows } P (a \# e)$
 $\langle \text{proof} \rangle$

lemma *not-knows-conc-knows-elem-not-knows-tail*:
assumes $\neg \text{knows } P (a \# e)$
and $\text{knows } P [a]$
shows $\neg \text{knows } P e$
 $\langle \text{proof} \rangle$

lemma *not-knows-conc-not-knows-elem-tail*:
assumes $\neg \text{knows } P (a \# e)$
shows $\neg \text{knows } P [a] \vee \neg \text{knows } P e$
 $\langle \text{proof} \rangle$

lemma *not-knows-elem-not-knows-conc*:
assumes $\neg \text{knows } P [a]$
shows $\neg \text{knows } P (a \# e)$
 $\langle \text{proof} \rangle$

lemma *not-knows-tail-not-knows-conc*:
assumes $\neg \text{knows } P e$
shows $\neg \text{knows } P (a \# e)$
 $\langle \text{proof} \rangle$

lemma *knows-composition3*:
fixes $e::\text{Expression list}$
assumes $\text{knows } P e$
and $\text{subPQ:subcomponents } PQ = \{P, Q\}$


```

and cCompI:correctCompositionIn PQ
and cCompKS:correctCompositionKS PQ
and  $\forall (m::\text{Expression}). ((m \text{ mem } e) \longrightarrow$ 
     $((\exists z1. m = (kE z1)) \vee (\exists z2. m = (sE z2))))$ 
and notSpecKeysSecretsExpr P e
and notSpecKeysSecretsExpr Q e
shows knows PQ e
<proof>

```

```

lemma knows-composition4:
assumes knows Q e
and subPQ:subcomponents PQ = {P,Q}
and cCompI:correctCompositionIn PQ
and cCompKS:correctCompositionKS PQ
and  $\forall m. m \text{ mem } e \longrightarrow ((\exists z. m = kE z) \vee (\exists z. m = sE z))$ 
and notSpecKeysSecretsExpr P e
and notSpecKeysSecretsExpr Q e
shows knows PQ e
<proof>

```

```

lemma knows-composition5:
assumes knows P e  $\vee$  knows Q e
and subcomponents PQ = {P,Q}
and correctCompositionIn PQ
and correctCompositionKS PQ
and  $\forall m. m \text{ mem } e \longrightarrow ((\exists z. m = kE z) \vee (\exists z. m = sE z))$ 
and notSpecKeysSecretsExpr P e
and notSpecKeysSecretsExpr Q e
shows knows PQ e
<proof>

```

end

References

- [1] T. Nipkow, L. C. Paulson, and M. Wenzel. *Isabelle/HOL – A Proof Assistant for Higher-Order Logic*. LNCS. Springer, 2013.
- [2] M. Spichkova. Stream Processing Components: Isabelle/HOL Formalisation and Case Studies. *Archive of Formal Proofs*, Nov. 2013.
- [3] M. Spichkova and J. Jürjens. Formal Specification of Cryptographic Protocols and Their Composition Properties: FOCUS-oriented approach. Technical report, Technische Universität München, 2008.