

Theorem: Simple4WHS

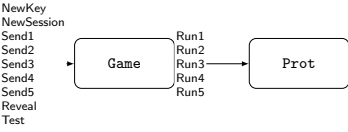
September 30, 2025

Contents

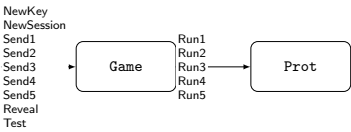
1	Games	2
1.1	KX Game	3
1.2	KX_NoKey Game	5
1.3	KX_NoPrf_RealPrf Game	7
1.4	KX_NoPrf_IdealPrf Game	9
1.5	RealKX_NoPrf_IdealPrf Game	11
1.6	IdealKX_NoPrf_IdealPrf Game	13
1.7	PRFideal Game	15
1.8	PRFreal Game	16
2	Advantages	17
3	Gamehops	17
3.1	Equivalence between KX and KX_NoKey	17
3.2	Equivalence between KX_NoKey and KX_NoPrf_RealPrf	17
3.3	Reduction to prf	17
3.3.1	Game KX_NoPrf_RealPrf with Assumption Game PRFreal highlighted in red	17
3.3.2	Game KX_NoPrf_IdealPrf with Assumption Game PRFideal highlighted in red	17
3.4	Equivalence between RealKX_NoPrf_IdealPrf and IdealKX_NoPrf_IdealPrf	17

1 Games

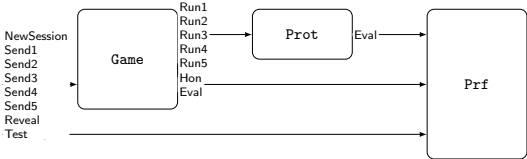
KX Game



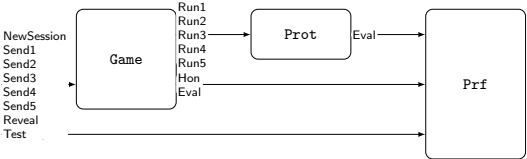
KX_NoKey Game



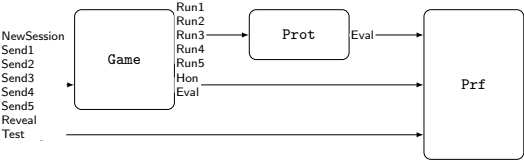
KX_NoPrf_RealPrf Game



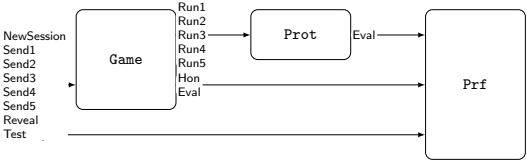
KX_NoPrf_IdealPrf Game



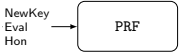
RealKX_NoPrf_IdealPrf Game



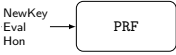
IdealKX_NoPrf_IdealPrf Game



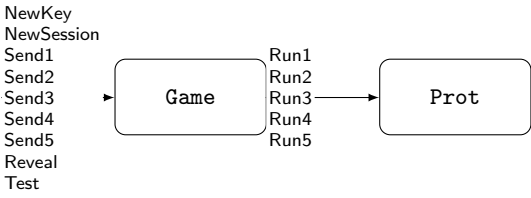
PRFideal Game



PRFreal Game



1.1 KX Game



```

Prot
Run1(state)


---


parse state as  $\left( \begin{array}{c} U, u, V, ltk, acc, k, \\ ni, nr, kmac, sid, \\ mess \end{array} \right)$ 

assert u = false
assert acc =  $\perp$ 
assert mess = 0

 $ni \stackrel{id2}{\leftarrow} \{0, 1\}^{256}$ 

return  $\left( \begin{array}{c} \left( \begin{array}{c} U, u, V, ltk, \perp, k, ni, \\ nr, kmac, sid, 1 \end{array} \right), \\ (ni) \end{array} \right)$ 

```

```

Run2(state, ni)
  parse state as  $\begin{pmatrix} U, v, V, ltk, acc, \\ k, ni, \\ nr, kmac, \\ sid, mess \end{pmatrix}$ 
  assert v = true
  assert acc =  $\perp$ 
  assert mess = 0
   $nr \stackrel{id3}{\leftarrow} \{0, 1\}^{256}$ 
  kmac  $\leftarrow$  prf(ltk, U, V, ni, nr, false)
  k  $\leftarrow$  prf(ltk, U, V, ni, nr, true)
  tau  $\leftarrow$  mac(kmac, nr, 2)
  sid  $\leftarrow$  ( U, V, ni, nr, tau )
  return  $\begin{pmatrix} \begin{pmatrix} U, v, V, ltk, \perp, k, ni, \\ nr, kmac, sid, 1 \end{pmatrix}, \\ (nr, tau) \end{pmatrix}$ 

```

```

Run3(state, msg)


---


parse state as  $\left( \begin{array}{c} U, u, V, ltk, acc, \\ k, ni, nr, \\ kmac, \\ sid, mess \end{array} \right)$ 

assert u = false
assert acc =  $\perp$ 
assert mess = 1

parse msg as ( nr, tau )

kmac  $\leftarrow$  prf(ltk, U, V, ni, nr, false)
k  $\leftarrow$  prf(ltk, U, V, ni, nr, true)
tau  $\leftarrow$  mac(kmac, ni, 3)
sid  $\leftarrow$  ( U, V, ni, nr, tau )

if mac(kmac, nr, 2) = tau then

    return  $\left( \begin{array}{c} \left( \begin{array}{c} U, u, V, ltk, \perp, k, ni, \\ nr, kmac, sid, 2 \end{array} \right), \\ (ni, tau) \end{array} \right)$ 

else

    return  $\left( \begin{array}{c} state, \\ (zeron, zeron) \end{array} \right)$ 

```

```

Run4(state, msg)
  parse state as  $\left( \begin{array}{c} U, v, V, ltk, acc, k, \\ ni, nr, kmac, sid, \\ mess \end{array} \right)$ 
  assert v = true
  assert acc = ⊥
  assert mess = 1
  parse msg as  $\left( \begin{array}{c} ni, tau \end{array} \right)$ 
  if  $\left( \begin{array}{c} mac(kmac, ni, 3) = tau \\ \wedge ni = ni \end{array} \right)$  then
    acc ← true
    tau ← mac(kmac, zeron, 4)
  return  $\left( \begin{array}{c} \left( \begin{array}{c} U, v, V, ltk, acc, k, \\ ni, nr, kmac, sid, 2 \end{array} \right), \\ tau \end{array} \right)$ 
else
  acc ← false
  return  $\left( \begin{array}{c} \left( \begin{array}{c} U, v, V, ltk, acc, k, \\ ni, nr, kmac, sid, 2 \end{array} \right), \\ zeron \end{array} \right)$ 

```

```

Run5(state, tau)
-----
parse state as  $\left( \begin{array}{l} U, u, V, ltk, acc, k, ni, \\ nr, kmac, sid, mess \end{array} \right)$ 
assert u = false
assert acc =  $\perp$ 
assert mess = 2
if mac(kmac, zeron, 4) = tau then
    (

```

Game
NewKey(ltk)

```

if  $ltk \neq \perp$  then
   $ltk \stackrel{id1}{\leftarrow} \{0, 1\}^{256}$ 
   $LTK[kid] \leftarrow ltk$ 
   $H[kid] \leftarrow \text{true}$ 
else
   $LTK[kid] \leftarrow ltk$ 
   $H[kid] \leftarrow \text{false}$ 
return  $kid$ 

```

```

NewSession( $U, u, V, kid$ )


---


assert ( $LTK[kid] \neq \perp$ )
 $ctr \leftarrow (ctr + 1)$ 
 $ltk \leftarrow LTK[kid]$ 
 $State[ctr] \leftarrow \left( \begin{array}{c} U, u, V, ltk, \perp, \perp, \perp, \perp, \\ \perp, \perp, 0 \end{array} \right)$ 
 $Fresh[ctr] \leftarrow H[kid]$ 
return  $ctr$ 

```

```

Send1(ctr)
-----
assert (State[ctr]  $\neq \perp$ )
state  $\leftarrow$  State[ctr]
return  $\stackrel{\text{invoke}}{\leftarrow}$  Run1(state) // Pkg: Prot
parse return as ( state, msg )
State[ctr]  $\leftarrow$  state
return msg

```

$$\begin{array}{l} \text{Send2}(ctr, msg) \\ \hline \mathbf{assert} \ (State[ctr] \neq \perp) \\ state \leftarrow State[ctr] \\ \text{return} \stackrel{\text{invoke}}{\leftarrow} \text{Run2}(state, msg) \quad // \text{ Pkg: Prot} \\ \mathbf{parse} \ \text{return} \ \mathbf{as} \ (\ state, msg \) \\ State[ctr] \leftarrow state \\ \mathbf{return} \ msg \end{array}$$
$$\begin{array}{l} \text{Send3}(ctr, msg) \\ \hline \mathbf{assert} \ (State[ctr] \neq \perp) \\ state \leftarrow State[ctr] \\ \text{return} \stackrel{\text{invoke}}{\leftarrow} \text{Run3}(state, msg) \quad // \text{ Pkg: Prot} \\ \mathbf{parse} \ \text{return} \ \mathbf{as} \ (\ state, msg \) \\ State[ctr] \leftarrow state \\ \mathbf{return} \ msg \end{array}$$

```

Send4(ctr, msg)
-----
assert (State[ctr]  $\neq \perp$ )
state  $\leftarrow$  State[ctr]
return  $\stackrel{\text{invoke}}{\leftarrow}$  Run4(state, msg) // Pkg: Prot
parse return as ( state, msg )
State[ctr]  $\leftarrow$  state
return msg

```

```

Send5(ctr, msg)
assert (State[ctr]  $\neq \perp$ )
state  $\leftarrow$  State[ctr]
return  $\stackrel{\text{invoke}}{\leftarrow}$  Run5(state, msg) // Pkg: Prot
parse return as ( state, stop )
State[ctr]  $\leftarrow$  state
return stop

```

```

Reveal(ctr)

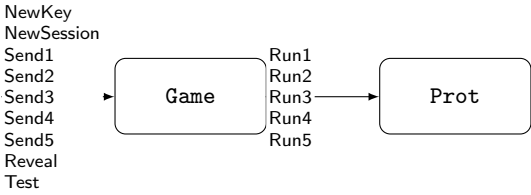

---


parse State[ctr] as  $\left( \begin{array}{c} U, u, V, ltk, acc, k, ni, \\ nr, kmac, sid, mess \end{array} \right)$ 
assert acc = true
assert RevTested[sid] =  $\perp$ 
RevTested[sid]  $\leftarrow$  false
return k

```

Test (<i>ctr</i>)	
<hr/>	
parse <i>State</i> [<i>ctr</i>] as	$\left(\begin{array}{c} U, u, V, ltk, acc, k, ni, \\ nr, kmac, sid, mess \end{array} \right)$
assert <i>acc</i> = true	
assert <i>Fresh</i> [<i>ctr</i>]	
assert <i>RevTested</i> [<i>sid</i>] = \perp	

1.2 KX_NoKey Game



```

Prot
Run1(state)


---


parse state as  $\left( \begin{array}{c} U, u, V, ltk, acc, \\ ni, nr, kmac, sid, \\ mess \end{array} \right)$ 

assert  $u = \text{false}$ 

assert  $acc = \perp$ 

assert  $mess = 0$ 

 $ni \stackrel{id2}{\leftarrow} \{0, 1\}^{256}$ 

return  $\left( \begin{array}{c} \left( \begin{array}{c} U, u, V, ltk, \perp, ni, nr, \\ kmac, sid, 1 \end{array} \right), \\ (ni) \end{array} \right)$ 

```

```

Run2(state, ni)


---


parse state as  $\left( \begin{array}{c} U, v, V, ltk, acc, \\ ni, nr, \\ kmac, \\ sid, mess \end{array} \right)$ 

assert  $v = \text{true}$ 

assert  $acc = \perp$ 

assert  $mess = 0$ 

 $nr \stackrel{id3}{\leftarrow} \{0, 1\}^{256}$ 

 $kmac \leftarrow \text{prf}(ltk, U, V, ni, nr, \text{false})$ 

 $\tau \leftarrow \text{mac}(kmac, nr, 2)$ 

 $sid \leftarrow (U, V, ni, nr, \tau)$ 

return  $\left( \begin{pmatrix} U, v, V, ltk, \perp, ni, nr, \\ kmac, sid, 1 \\ nr, \tau \end{pmatrix} \right)$ 

```

```

Run3(state, msg)


---


parse state as  $\left( \begin{array}{c} U, u, V, ltk, acc, ni, \\ nr, kmac, \\ sid, mess \end{array} \right)$ 

assert u = false

assert acc =  $\perp$ 

assert mess = 1

parse msg as  $(nr, \tau)$ 

kmac  $\leftarrow \text{prf}(ltk, U, V, ni, nr, \text{false})$ 

tau  $\leftarrow \text{mac}(kmac, ni, 3)$ 

sid  $\leftarrow (U, V, ni, nr, \tau)$ 

if  $\text{mac}(kmac, nr, 2) = \tau$  then

    return  $\left( \left( \begin{array}{c} U, u, V, ltk, \perp, ni, nr, \\ kmac, sid, 2 \\ (ni, \tau) \end{array} \right), \right)$ 

else

    return  $\left( \left( \begin{array}{c} state, \\ (zeron, zeron) \end{array} \right) \right)$ 

```

```

Run4(state, msg)


---


parse state as  $\left( \begin{array}{c} U, v, V, ltk, acc, \\ ni, nr, kmac, sid, \\ mess \end{array} \right)$ 

assert  $v = \text{true}$ 
assert  $acc = \perp$ 
assert  $mess = 1$ 

parse msg as  $(ni, \tau)$ 

if  $\left( \begin{array}{c} mac(kmac, ni, 3) = \tau \\ \wedge ni = ni \end{array} \right)$  then

   $acc \leftarrow \text{true}$ 
   $\tau \leftarrow mac(kmac, zeron, 4)$ 

return  $\left( \begin{array}{c} \left( \begin{array}{c} U, v, V, ltk, acc, \\ ni, nr, kmac, sid, 2 \end{array} \right), \\ (\tau) \end{array} \right)$ 

else

   $acc \leftarrow \text{false}$ 

return  $\left( \begin{array}{c} \left( \begin{array}{c} U, v, V, ltk, acc, \\ ni, nr, kmac, sid, 2 \end{array} \right), \\ (zeron) \end{array} \right)$ 

```

```

Run5(state, tau)


---


parse state as  $\left( \begin{array}{c} U, u, V, ltk, acc, ni, nr, \\ kmac, sid, mess \end{array} \right)$ 
assert u = false
assert acc =  $\perp$ 
assert mess = 2
if mac(kmac, zeron, 4) = tau then
  return  $\left( \begin{array}{c} \left( \begin{array}{c} U, u, V, ltk, \text{true}, \\ ni, nr, kmac, sid, 3 \end{array} \right), \\ \left( \text{true} \right) \end{array} \right)$ 

```

```

Game
NewKey(ltk)


---


kid  $\leftarrow (kid + 1)$ 
if ltk =  $\perp$  then
  ltk  $\xleftarrow{\text{id1}} \{0, 1\}^{256}$ 
  LTK[kid]  $\leftarrow ltk$ 
  H[kid]  $\leftarrow \text{true}$ 
else
  LTK[kid]  $\leftarrow ltk$ 
  H[kid]  $\leftarrow \text{false}$ 
return kid

NewSession(U, u, V, kid)


---


assert (LTK[kid]  $\neq \perp$ )
ctr  $\leftarrow (ctr + 1)$ 
ltk  $\leftarrow LTK[kid]$ 
State[ctr]  $\leftarrow \left( \begin{array}{c} U, u, V, ltk, \perp, \perp, \perp, \perp, \\ \perp, 0 \end{array} \right)$ 
Fresh[ctr]  $\leftarrow H[kid]$ 
return ctr

```

```

Send1(ctr)
-----
assert (State[ctr]  $\neq \perp$ )
state  $\leftarrow$  State[ctr]
return  $\stackrel{\text{invoke}}{\leftarrow}$  Run1(state) // Pkg: Prot
parse return as ( state, msg )
State[ctr]  $\leftarrow$  state
return msg

```

```

Send2(ctr, msg)


---


assert (State[ctr]  $\neq \perp$ )
state  $\leftarrow$  State[ctr]
return  $\stackrel{\text{invoke}}{\leftarrow}$  Run2(state, msg) // Pkg: Prot
parse return as ( state, msg )
State[ctr]  $\leftarrow$  state
return msg

Send3(ctr, msg)


---


assert (State[ctr]  $\neq \perp$ )
state  $\leftarrow$  State[ctr]
return  $\stackrel{\text{invoke}}{\leftarrow}$  Run3(state, msg) // Pkg: Prot
parse return as ( state, msg )
State[ctr]  $\leftarrow$  state
return msg

```

```

Send4(ctr, msg)


---


assert (State[ctr]  $\neq \perp$ )
state  $\leftarrow$  State[ctr]
return  $\stackrel{\text{invoke}}{\leftarrow}$  Run4(state, msg) // Pkg: Prot
parse return as ( state, msg )
State[ctr]  $\leftarrow$  state
return msg

```

```

Send5(ctr, msg)
assert (State[ctr]  $\neq \perp$ )
state  $\leftarrow$  State[ctr]
return  $\stackrel{\text{invoke}}{\leftarrow}$  Run5(state, msg) // Pkg: Prot
parse return as ( state, stop )
State[ctr]  $\leftarrow$  state
return stop

```

```

Reveal(ctr)
parse State[ctr] as  $\left( \begin{array}{c} U, u, V, ltk, acc, ni, nr, \\ kmac, sid, mess \end{array} \right)$ 
assert acc = true
assert RevTested[sid] =  $\perp$ 
RevTested[sid]  $\leftarrow$  false
k  $\leftarrow$  prf(ltk, U, V, ni, nr, true)
return k

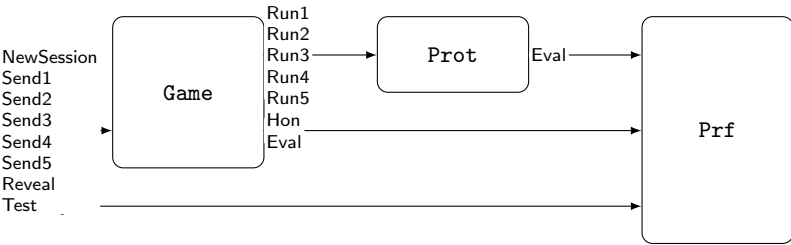
```

```

Test(ctr)
parse State[ctr] as  $\left( \begin{array}{c} U, u, V, ltk, acc, ni, nr, \\ kmac, sid, mess \end{array} \right)$ 
assert acc = true
assert Fresh[ctr]

```

1.3 KX_NoPrf_RealPrf Game



```

Prf
NewKey(ltk)

kid ← (kid + 1)
if ltk = ⊥ then
    ltk  $\xleftarrow{\$}$  {0, 1}256
    LTk[kid] ← ltk
    H[kid] ← true
else
    LTk[kid] ← ltk
    H[kid] ← false
return kid

```

```

Eval(h, a, aa, c, d, e)
assert (LTk[h] ≠ ⊥)
if (H[h] = false ∨ ¬b) then
    k ← LTk[h]
    return prf(k, a, aa, c, d, e)
if PRF[( h, a, aa, c, d, e )] = ⊥ then
    temp  $\xleftarrow{\$}$  {0, 1}256
    PRF[( h, a, aa, c, d, e )] ← temp
    return temp
y ← PRF[( h, a, aa, c, d, e )]
return y

```

```

Hon(h)
assert (H[h] ≠ ⊥)
return H[h]

```

```

Prot
Run1(state)

parse state as  $\left( \begin{array}{c} U, u, V, ltk, acc, \\ ni, nr, kmac, sid, \\ mess \end{array} \right)$ 

assert u = false
assert acc = ⊥
assert mess = 0
 $ni \xleftarrow{\$} \{0, 1\}^{256}$ 

return  $\left( \begin{array}{c} \left( \begin{array}{c} U, u, V, ltk, \perp, ni, nr, \\ kmac, sid, 1 \\ (ni) \end{array} \right), \end{array} \right)$ 

```

```

Run2(state, ni)

parse state as  $\left( \begin{array}{c} U, v, V, ltk, acc, \\ ni, nr, \\ kmac, \\ sid, mess \end{array} \right)$ 

assert v = true
assert acc = ⊥
assert mess = 0
 $nr \xleftarrow{\$} \{0, 1\}^{256}$ 
 $kmac \xleftarrow{\text{invoke}} \text{Eval}(ltk, U, V, ni, nr, \text{false}) \quad // \text{ Pkg: Prf}$ 
tau ← mac(kmac, nr, 2)
sid ← ( U, V, ni, nr, tau )

return  $\left( \begin{array}{c} \left( \begin{array}{c} U, v, V, ltk, \perp, ni, nr, \\ kmac, sid, 1 \\ (nr, tau) \end{array} \right), \end{array} \right)$ 

```

```

Run3(state, msg)

parse state as  $\left( \begin{array}{c} U, u, V, ltk, acc, ni, \\ nr, kmac, \\ sid, mess \end{array} \right)$ 

assert u = false
assert acc = ⊥
assert mess = 1
parse msg as ( nr, tau )

 $kmac \xleftarrow{\text{invoke}} \text{Eval}(ltk, U, V, ni, nr, \text{false}) \quad // \text{ Pkg: Prf}$ 
tau ← mac(kmac, ni, 3)
sid ← ( U, V, ni, nr, tau )
if mac(kmac, nr, 2) = tau then

    return  $\left( \begin{array}{c} \left( \begin{array}{c} U, u, V, ltk, \perp, ni, nr, \\ kmac, sid, 2 \\ (ni, tau) \end{array} \right), \end{array} \right)$ 

else

    return  $\left( \begin{array}{c} state, \\ ( zeron, zeron ) \end{array} \right)$ 

```

```

Run4(state, msg)

parse state as  $\left( \begin{array}{c} U, v, V, ltk, acc, \\ ni, nr, kmac, sid, \\ mess \end{array} \right)$ 

assert v = true
assert acc = ⊥
assert mess = 1
parse msg as ( ni, tau )

if  $\left( \begin{array}{c} mac(kmac, ni, 3) = tau \\ \wedge ni = ni \end{array} \right)$  then
    acc ← true
    tau ← mac(kmac, zeron, 4)

    return  $\left( \begin{array}{c} \left( \begin{array}{c} U, v, V, ltk, acc, \\ ni, nr, kmac, sid, 2 \\ (tau) \end{array} \right), \end{array} \right)$ 

else
    acc ← false

    return  $\left( \begin{array}{c} \left( \begin{array}{c} U, v, V, ltk, acc, \\ ni, nr, kmac, sid, 2 \\ ( zeron ) \end{array} \right), \end{array} \right)$ 

```

```

Run5(state, tau)

parse state as  $\left( \begin{array}{c} U, u, V, ltk, acc, ni, nr, \\ kmac, sid, mess \end{array} \right)$ 

assert u = false
assert acc = ⊥
assert mess = 2
if mac(kmac, zeron, 4) = tau then

    return  $\left( \begin{array}{c} \left( \begin{array}{c} U, u, V, ltk, \text{true}, \\ ni, nr, kmac, sid, 3 \\ ( zeron ) \end{array} \right), \end{array} \right)$ 

```

```

Game
NewSession(U, u, V, kid)

ctr ← (ctr + 1)
hon  $\xleftarrow{\text{invoke}} \text{Hon}(kid) \quad // \text{ Pkg: Prf}$ 

State[ctr] ←  $\left( \begin{array}{c} U, u, V, kid, \perp, \perp, \perp, \perp, \\ \perp, 0 \end{array} \right)$ 

Fresh[ctr] ← hon
return ctr

```

```

Send1(ctr)

assert (State[ctr] ≠ ⊥)
state ← State[ctr]

return  $\xleftarrow{\text{invoke}} \text{Run1}(state) \quad // \text{ Pkg: Prot}$ 

parse return as ( state, msg )
State[ctr] ← state
return msg

```

```

Send2(ctr, msg)

assert (State[ctr] ≠ ⊥)
state ← State[ctr]

return  $\xleftarrow{\text{invoke}} \text{Run2}(state, msg) \quad // \text{ Pkg: Prot}$ 

parse return as ( state, msg )
State[ctr] ← state
return msg

```

```

Send3(ctr, msg)

assert (State[ctr] ≠ ⊥)
state ← State[ctr]

return  $\xleftarrow{\text{invoke}} \text{Run3}(state, msg) \quad // \text{ Pkg: Prot}$ 

parse return as ( state, msg )
State[ctr] ← state
return msg

```

```

Send4(ctr, msg)

assert (State[ctr] ≠ ⊥)
state ← State[ctr]

return  $\xleftarrow{\text{invoke}} \text{Run4}(state, msg) \quad // \text{ Pkg: Prot}$ 

parse return as ( state, msg )
State[ctr] ← state
return msg

```

```

Send5(ctr, msg)

assert (State[ctr] ≠ ⊥)
state ← State[ctr]

return  $\xleftarrow{\text{invoke}} \text{Run5}(state, msg) \quad // \text{ Pkg: Prot}$ 

parse return as ( state, stop )
State[ctr] ← state
return stop

```

```

Reveal(ctr)

parse State[ctr] as  $\left( \begin{array}{c} U, u, V, ltk, acc, ni, nr, \\ kmac, sid, mess \end{array} \right)$ 

assert acc = true
assert RevTested[sid] = ⊥
RevTested[sid] ← false
 $k \xleftarrow{\text{invoke}} \text{Eval}(ltk, U, V, ni, nr, \text{true}) \quad // \text{ Pkg: Prf}$ 
return k

```

```

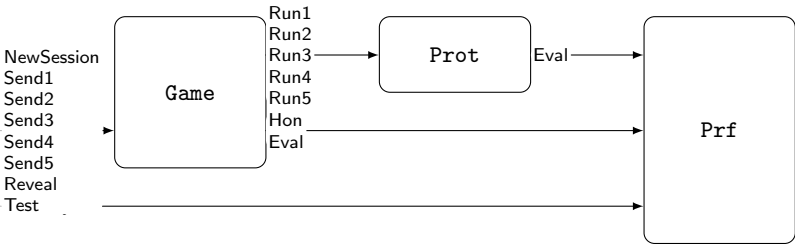
Test(ctr)

parse State[ctr] as  $\left( \begin{array}{c} U, u, V, ltk, acc, ni, nr, \\ kmac, sid, mess \end{array} \right)$ 

assert acc = true
assert Fresh[ctr]
assert RevTested[sid] = ⊥
RevTested[sid] ← true
if b then
     $k \xleftarrow{\$} \{0, 1\}^{256}$ 
else
     $k \xleftarrow{\text{invoke}} \text{Eval}(ltk, U, V, ni, nr, \text{true}) \quad // \text{ Pkg: Prf}$ 
return k

```


1.4 KX_NoPrf_IdealPrf Game



Prf

NewKey(ltk)

$kid \leftarrow (kid + 1)$

if $ltk = \perp$ **then**

$ltk \xleftarrow{\$ \text{id1}} \{0, 1\}^{256}$
 $LTk[kid] \leftarrow ltk$
 $H[kid] \leftarrow \text{true}$

else

$LTk[kid] \leftarrow ltk$
 $H[kid] \leftarrow \text{false}$

return kid

Eval(h, a, aa, c, d, e)

assert $(LTk[h] \neq \perp)$

if $(H[h] = \text{false} \vee \neg b)$ **then**

$k \leftarrow LTk[h]$
return $prf(k, a, aa, c, d, e)$

if $PRF[(\ h, a, aa, c, d, e\)] = \perp$ **then**

$temp \xleftarrow{\$ \text{id5}} \{0, 1\}^{256}$
 $PRF[(\ h, a, aa, c, d, e\)] \leftarrow temp$
return $temp$

$y \leftarrow PRF[(\ h, a, aa, c, d, e\)]$
return y

Hon(h)

assert $(H[h] \neq \perp)$
return $H[h]$

Prot

Run1($state$)

parse $state$ **as** $\left(\begin{array}{c} U, u, V, ltk, acc, \\ ni, nr, kmac, sid, \\ mess \end{array}\right)$

assert $u = \text{false}$
assert $acc = \perp$
assert $mess = 0$
 $ni \xleftarrow{\$ \text{id2}} \{0, 1\}^{256}$

return $\left(\begin{array}{c} \left(\begin{array}{c} U, u, V, ltk, \perp, ni, nr, \\ kmac, sid, 1 \\ (ni) \end{array}\right), \end{array}\right)$

Run2($state, ni$)

parse $state$ **as** $\left(\begin{array}{c} U, v, V, ltk, acc, \\ ni, nr, \\ kmac, \\ sid, mess \end{array}\right)$

assert $v = \text{true}$
assert $acc = \perp$
assert $mess = 0$
 $nr \xleftarrow{\$ \text{id3}} \{0, 1\}^{256}$
 $kmac \xleftarrow{\text{invoke}} \text{Eval}(ltk, U, V, ni, nr, \text{false}) \quad // \text{ Pkg: Prf}$
 $\tau \leftarrow mac(kmac, nr, 2)$
 $sid \leftarrow (\ U, V, ni, nr, \tau \)$

return $\left(\begin{array}{c} \left(\begin{array}{c} U, v, V, ltk, \perp, ni, nr, \\ kmac, sid, 1 \\ (nr, \tau) \end{array}\right), \end{array}\right)$

Run3($state, msg$)

parse $state$ **as** $\left(\begin{array}{c} U, u, V, ltk, acc, ni, \\ nr, kmac, \\ sid, mess \end{array}\right)$

assert $u = \text{false}$
assert $acc = \perp$
assert $mess = 1$
parse msg **as** $(\ nr, \tau \)$
 $kmac \xleftarrow{\text{invoke}} \text{Eval}(ltk, U, V, ni, nr, \text{false}) \quad // \text{ Pkg: Prf}$
 $\tau \leftarrow mac(kmac, ni, 3)$
 $sid \leftarrow (\ U, V, ni, nr, \tau \)$
if $mac(kmac, nr, 2) = \tau$ **then**

return $\left(\begin{array}{c} \left(\begin{array}{c} U, u, V, ltk, \perp, ni, nr, \\ kmac, sid, 2 \\ (ni, \tau) \end{array}\right), \end{array}\right)$

else

return $\left(\begin{array}{c} state, \\ (zeron, zeron) \end{array}\right)$

Run4($state, msg$)

parse $state$ **as** $\left(\begin{array}{c} U, v, V, ltk, acc, \\ ni, nr, kmac, sid, \\ mess \end{array}\right)$

assert $v = \text{true}$
assert $acc = \perp$
assert $mess = 1$
parse msg **as** $(\ ni, \tau \)$
if $\left(\begin{array}{c} mac(kmac, ni, 3) = \tau \\ \wedge ni = ni \end{array}\right)$ **then**

$acc \leftarrow \text{true}$
 $\tau \leftarrow mac(kmac, zeron, 4)$

return $\left(\begin{array}{c} \left(\begin{array}{c} U, v, V, ltk, acc, \\ ni, nr, kmac, sid, 2 \\ (\tau) \end{array}\right), \end{array}\right)$

else

$acc \leftarrow \text{false}$

return $\left(\begin{array}{c} \left(\begin{array}{c} U, v, V, ltk, acc, \\ ni, nr, kmac, sid, 2 \\ (zeron) \end{array}\right), \end{array}\right)$

Run5($state, \tau$)

parse $state$ **as** $\left(\begin{array}{c} U, u, V, ltk, acc, ni, nr, \\ kmac, sid, mess \end{array}\right)$

assert $u = \text{false}$
assert $acc = \perp$
assert $mess = 2$
if $mac(kmac, zeron, 4) = \tau$ **then**

return $\left(\begin{array}{c} \left(\begin{array}{c} U, u, V, ltk, \text{true}, \\ ni, nr, kmac, sid, 3 \\ (\tau) \end{array}\right), \end{array}\right)$

Game

NewSession(U, u, V, kid)

$ctr \leftarrow (ctr + 1)$
 $hon \xleftarrow{\text{invoke}} \text{Hon}(kid) \quad // \text{ Pkg: Prf}$

$State[ctr] \leftarrow \left(\begin{array}{c} U, u, V, kid, \perp, \perp, \perp, \perp, \\ \perp, 0 \end{array}\right)$

$Fresh[ctr] \leftarrow hon$

return ctr

Send1(ctr)

assert $(State[ctr] \neq \perp)$
 $state \leftarrow State[ctr]$

$return \xleftarrow{\text{invoke}} \text{Run1}(state) \quad // \text{ Pkg: Prot}$

parse $return$ **as** $(\ state, msg \)$
 $State[ctr] \leftarrow state$
return msg

Send2(ctr, msg)

assert $(State[ctr] \neq \perp)$
 $state \leftarrow State[ctr]$

$return \xleftarrow{\text{invoke}} \text{Run2}(state, msg) \quad // \text{ Pkg: Prot}$

parse $return$ **as** $(\ state, msg \)$
 $State[ctr] \leftarrow state$
return msg

Send3(ctr, msg)

assert $(State[ctr] \neq \perp)$
 $state \leftarrow State[ctr]$

$return \xleftarrow{\text{invoke}} \text{Run3}(state, msg) \quad // \text{ Pkg: Prot}$

parse $return$ **as** $(\ state, msg \)$
 $State[ctr] \leftarrow state$
return msg

Send4(ctr, msg)

assert $(State[ctr] \neq \perp)$
 $state \leftarrow State[ctr]$

$return \xleftarrow{\text{invoke}} \text{Run4}(state, msg) \quad // \text{ Pkg: Prot}$

parse $return$ **as** $(\ state, msg \)$
 $State[ctr] \leftarrow state$
return msg

Send5(ctr, msg)

assert $(State[ctr] \neq \perp)$
 $state \leftarrow State[ctr]$

$return \xleftarrow{\text{invoke}} \text{Run5}(state, msg) \quad // \text{ Pkg: Prot}$

parse $return$ **as** $(\ state, stop \)$
 $State[ctr] \leftarrow state$
return $stop$

Reveal(ctr)

parse $State[ctr]$ **as** $\left(\begin{array}{c} U, u, V, ltk, acc, ni, nr, \\ kmac, sid, mess \end{array}\right)$

assert $acc = \text{true}$
assert $RevTested[sid] = \perp$
 $RevTested[sid] \leftarrow \text{false}$
 $k \xleftarrow{\text{invoke}} \text{Eval}(ltk, U, V, ni, nr, \text{true}) \quad // \text{ Pkg: Prf}$
return k

Test(ctr)

parse $State[ctr]$ **as** $\left(\begin{array}{c} U, u, V, ltk, acc, ni, nr, \\ kmac, sid, mess \end{array}\right)$

assert $acc = \text{true}$
assert $Fresh[ctr]$
assert $RevTested[sid] = \perp$
 $RevTested[sid] \leftarrow \text{true}$
if b **then**

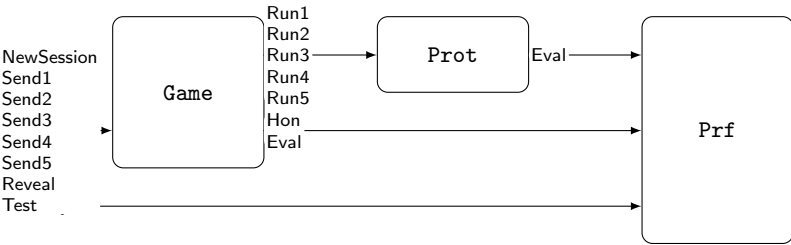
$k \xleftarrow{\$ \text{id4}} \{0, 1\}^{256}$

else

$k \xleftarrow{\text{invoke}} \text{Eval}(ltk, U, V, ni, nr, \text{true}) \quad // \text{ Pkg: Prf}$

return k

1.5 RealKX_NoPrf_IdealPrf Game



```

Prf
NewKey(ltk)

kid ← (kid + 1)
if ltk = ⊥ then
    ltk  $\xleftarrow{\$}$  {0, 1}256
    LTk[kid] ← ltk
    H[kid] ← true
else
    LTk[kid] ← ltk
    H[kid] ← false
return kid

```

```

Eval(h, a, aa, c, d, e)
assert (LTk[h] ≠ ⊥)
if (H[h] = false ∨ ¬b) then
    k ← LTk[h]
    return prf(k, a, aa, c, d, e)
if PRF[( h, a, aa, c, d, e )] = ⊥ then
    temp  $\xleftarrow{\$}$  {0, 1}256
    PRF[( h, a, aa, c, d, e )] ← temp
    return temp
y ← PRF[( h, a, aa, c, d, e )]
return y

```

```

Hon(h)
assert (H[h] ≠ ⊥)
return H[h]

```

```

Prot
Run1(state)

parse state as  $\left( \begin{array}{c} U, u, V, ltk, acc, \\ ni, nr, kmac, sid, \\ mess \end{array} \right)$ 

assert u = false
assert acc = ⊥
assert mess = 0
 $ni \xleftarrow{\$} \{0, 1\}^{256}$ 

return  $\left( \begin{array}{c} \left( \begin{array}{c} U, u, V, ltk, \perp, ni, nr, \\ kmac, sid, 1 \\ (ni) \end{array} \right), \end{array} \right)$ 

```

```

Run2(state, ni)

parse state as  $\left( \begin{array}{c} U, v, V, ltk, acc, \\ ni, nr, \\ kmac, \\ sid, mess \end{array} \right)$ 

assert v = true
assert acc = ⊥
assert mess = 0
 $nr \xleftarrow{\$} \{0, 1\}^{256}$ 
 $kmac \xleftarrow{\text{invoke}} \text{Eval}(ltk, U, V, ni, nr, \text{false}) \quad // \text{ Pkg: Prf}$ 
tau ← mac(kmac, nr, 2)
sid ← ( U, V, ni, nr, tau )

return  $\left( \begin{array}{c} \left( \begin{array}{c} U, v, V, ltk, \perp, ni, nr, \\ kmac, sid, 1 \\ (nr, tau) \end{array} \right), \end{array} \right)$ 

```

```

Run3(state, msg)

parse state as  $\left( \begin{array}{c} U, u, V, ltk, acc, ni, \\ nr, kmac, \\ sid, mess \end{array} \right)$ 

assert u = false
assert acc = ⊥
assert mess = 1
parse msg as ( nr, tau )

 $kmac \xleftarrow{\text{invoke}} \text{Eval}(ltk, U, V, ni, nr, \text{false}) \quad // \text{ Pkg: Prf}$ 
tau ← mac(kmac, ni, 3)
sid ← ( U, V, ni, nr, tau )
if mac(kmac, nr, 2) = tau then

    return  $\left( \begin{array}{c} \left( \begin{array}{c} U, u, V, ltk, \perp, ni, nr, \\ kmac, sid, 2 \\ (ni, tau) \end{array} \right), \end{array} \right)$ 

else

    return  $\left( \begin{array}{c} state, \\ ( zeron, zeron ) \end{array} \right)$ 

```

```

Run4(state, msg)

parse state as  $\left( \begin{array}{c} U, v, V, ltk, acc, \\ ni, nr, kmac, sid, \\ mess \end{array} \right)$ 

assert v = true
assert acc = ⊥
assert mess = 1
parse msg as ( ni, tau )

if  $\left( \begin{array}{c} mac(kmac, ni, 3) = tau \\ \wedge ni = ni \end{array} \right)$  then
    acc ← true
    tau ← mac(kmac, zeron, 4)

    return  $\left( \begin{array}{c} \left( \begin{array}{c} U, v, V, ltk, acc, \\ ni, nr, kmac, sid, 2 \\ (tau) \end{array} \right), \end{array} \right)$ 

else
    acc ← false

    return  $\left( \begin{array}{c} \left( \begin{array}{c} U, v, V, ltk, acc, \\ ni, nr, kmac, sid, 2 \\ ( zeron ) \end{array} \right), \end{array} \right)$ 

```

```

Run5(state, tau)

parse state as  $\left( \begin{array}{c} U, u, V, ltk, acc, ni, nr, \\ kmac, sid, mess \end{array} \right)$ 

assert u = false
assert acc = ⊥
assert mess = 2
if mac(kmac, zeron, 4) = tau then

    return  $\left( \begin{array}{c} \left( \begin{array}{c} U, u, V, ltk, \text{true}, \\ ni, nr, kmac, sid, 3 \\ ( zeron ) \end{array} \right), \end{array} \right)$ 

```

```

Game
NewSession(U, u, V, kid)

ctr ← (ctr + 1)
hon  $\xleftarrow{\text{invoke}} \text{Hon}(kid) \quad // \text{ Pkg: Prf}$ 

State[ctr] ←  $\left( \begin{array}{c} U, u, V, kid, \perp, \perp, \perp, \perp, \\ \perp, 0 \end{array} \right)$ 

Fresh[ctr] ← hon
return ctr

```

```

Send1(ctr)

assert (State[ctr] ≠ ⊥)
state ← State[ctr]

return  $\xleftarrow{\text{invoke}} \text{Run1}(state) \quad // \text{ Pkg: Prot}$ 

parse return as ( state, msg )
State[ctr] ← state
return msg

```

```

Send2(ctr, msg)

assert (State[ctr] ≠ ⊥)
state ← State[ctr]

return  $\xleftarrow{\text{invoke}} \text{Run2}(state, msg) \quad // \text{ Pkg: Prot}$ 

parse return as ( state, msg )
State[ctr] ← state
return msg

```

```

Send3(ctr, msg)

assert (State[ctr] ≠ ⊥)
state ← State[ctr]

return  $\xleftarrow{\text{invoke}} \text{Run3}(state, msg) \quad // \text{ Pkg: Prot}$ 

parse return as ( state, msg )
State[ctr] ← state
return msg

```

```

Send4(ctr, msg)

assert (State[ctr] ≠ ⊥)
state ← State[ctr]

return  $\xleftarrow{\text{invoke}} \text{Run4}(state, msg) \quad // \text{ Pkg: Prot}$ 

parse return as ( state, msg )
State[ctr] ← state
return msg

```

```

Send5(ctr, msg)

assert (State[ctr] ≠ ⊥)
state ← State[ctr]

return  $\xleftarrow{\text{invoke}} \text{Run5}(state, msg) \quad // \text{ Pkg: Prot}$ 

parse return as ( state, stop )
State[ctr] ← state
return stop

```

```

Reveal(ctr)

parse State[ctr] as  $\left( \begin{array}{c} U, u, V, ltk, acc, ni, nr, \\ kmac, sid, mess \end{array} \right)$ 

assert acc = true
assert RevTested[sid] = ⊥
RevTested[sid] ← false
 $k \xleftarrow{\text{invoke}} \text{Eval}(ltk, U, V, ni, nr, \text{true}) \quad // \text{ Pkg: Prf}$ 
return k

```

```

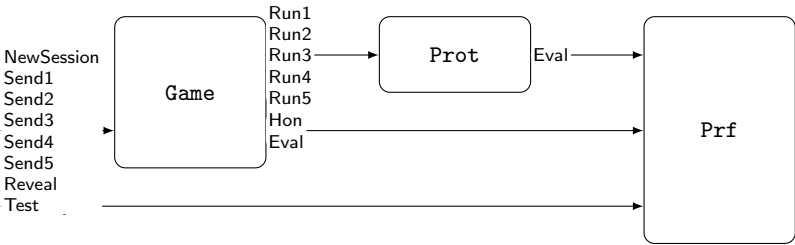
Test(ctr)

parse State[ctr] as  $\left( \begin{array}{c} U, u, V, ltk, acc, ni, nr, \\ kmac, sid, mess \end{array} \right)$ 

assert acc = true
assert Fresh[ctr]
assert RevTested[sid] = ⊥
RevTested[sid] ← true
if b then
     $k \xleftarrow{\$} \{0, 1\}^{256}$ 
else
     $k \xleftarrow{\text{invoke}} \text{Eval}(ltk, U, V, ni, nr, \text{true}) \quad // \text{ Pkg: Prf}$ 
return k

```

1.6 IdealKX_NoPrf_IdealPrf Game



```

Prf
NewKey(ltk)

kid ← (kid + 1)
if ltk = ⊥ then
    ltk  $\xleftarrow{\$}$  {0, 1}256
    LTk[kid] ← ltk
    H[kid] ← true
else
    LTk[kid] ← ltk
    H[kid] ← false
return kid

```

```

Eval(h, a, aa, c, d, e)
assert (LTk[h] ≠ ⊥)
if (H[h] = false ∨ ¬b) then
    k ← LTk[h]
    return prf(k, a, aa, c, d, e)
if PRF[( h, a, aa, c, d, e )] = ⊥ then
    temp  $\xleftarrow{\$}$  {0, 1}256
    PRF[( h, a, aa, c, d, e )] ← temp
    return temp
y ← PRF[( h, a, aa, c, d, e )]
return y

```

```

Hon(h)
assert (H[h] ≠ ⊥)
return H[h]

```

```

Prot
Run1(state)

parse state as  $\left( \begin{array}{c} U, u, V, ltk, acc, \\ ni, nr, kmac, sid, \\ mess \end{array} \right)$ 

assert u = false
assert acc = ⊥
assert mess = 0
 $ni \xleftarrow{\$} \{0, 1\}^{256}$ 

return  $\left( \begin{array}{c} \left( \begin{array}{c} U, u, V, ltk, \perp, ni, nr, \\ kmac, sid, 1 \\ (ni) \end{array} \right), \end{array} \right)$ 

```

```

Run2(state, ni)

parse state as  $\left( \begin{array}{c} U, v, V, ltk, acc, \\ ni, nr, \\ kmac, \\ sid, mess \end{array} \right)$ 

assert v = true
assert acc = ⊥
assert mess = 0
 $nr \xleftarrow{\$} \{0, 1\}^{256}$ 
 $kmac \xleftarrow{\text{invoke}} \text{Eval}(ltk, U, V, ni, nr, \text{false}) \quad // \text{ Pkg: Prf}$ 
tau ← mac(kmac, nr, 2)
sid ← ( U, V, ni, nr, tau )

return  $\left( \begin{array}{c} \left( \begin{array}{c} U, v, V, ltk, \perp, ni, nr, \\ kmac, sid, 1 \\ (nr, tau) \end{array} \right), \end{array} \right)$ 

```

```

Run3(state, msg)

parse state as  $\left( \begin{array}{c} U, u, V, ltk, acc, ni, \\ nr, kmac, \\ sid, mess \end{array} \right)$ 

assert u = false
assert acc = ⊥
assert mess = 1
parse msg as ( nr, tau )

 $kmac \xleftarrow{\text{invoke}} \text{Eval}(ltk, U, V, ni, nr, \text{false}) \quad // \text{ Pkg: Prf}$ 
tau ← mac(kmac, ni, 3)
sid ← ( U, V, ni, nr, tau )
if mac(kmac, nr, 2) = tau then

    return  $\left( \begin{array}{c} \left( \begin{array}{c} U, u, V, ltk, \perp, ni, nr, \\ kmac, sid, 2 \\ (ni, tau) \end{array} \right), \end{array} \right)$ 

else

    return  $\left( \begin{array}{c} state, \\ ( zeron, zeron ) \end{array} \right)$ 

```

```

Run4(state, msg)

parse state as  $\left( \begin{array}{c} U, v, V, ltk, acc, \\ ni, nr, kmac, sid, \\ mess \end{array} \right)$ 

assert v = true
assert acc = ⊥
assert mess = 1
parse msg as ( ni, tau )

if  $\left( \begin{array}{c} mac(kmac, ni, 3) = tau \\ \wedge ni = ni \end{array} \right)$  then
    acc ← true
    tau ← mac(kmac, zeron, 4)

    return  $\left( \begin{array}{c} \left( \begin{array}{c} U, v, V, ltk, acc, \\ ni, nr, kmac, sid, 2 \\ (tau) \end{array} \right), \end{array} \right)$ 

else
    acc ← false

    return  $\left( \begin{array}{c} \left( \begin{array}{c} U, v, V, ltk, acc, \\ ni, nr, kmac, sid, 2 \\ ( zeron ) \end{array} \right), \end{array} \right)$ 

```

```

Run5(state, tau)

parse state as  $\left( \begin{array}{c} U, u, V, ltk, acc, ni, nr, \\ kmac, sid, mess \end{array} \right)$ 

assert u = false
assert acc = ⊥
assert mess = 2
if mac(kmac, zeron, 4) = tau then

    return  $\left( \begin{array}{c} \left( \begin{array}{c} U, u, V, ltk, \text{true}, \\ ni, nr, kmac, sid, 3 \\ ( zeron ) \end{array} \right), \end{array} \right)$ 

```

```

Game
NewSession(U, u, V, kid)

ctr ← (ctr + 1)
hon  $\xleftarrow{\text{invoke}} \text{Hon}(kid) \quad // \text{ Pkg: Prf}$ 

State[ctr] ←  $\left( \begin{array}{c} U, u, V, kid, \perp, \perp, \perp, \perp, \\ \perp, 0 \end{array} \right)$ 

Fresh[ctr] ← hon
return ctr

```

```

Send1(ctr)

assert (State[ctr] ≠ ⊥)
state ← State[ctr]

return  $\xleftarrow{\text{invoke}} \text{Run1}(state) \quad // \text{ Pkg: Prot}$ 

parse return as ( state, msg )
State[ctr] ← state
return msg

```

```

Send2(ctr, msg)

assert (State[ctr] ≠ ⊥)
state ← State[ctr]

return  $\xleftarrow{\text{invoke}} \text{Run2}(state, msg) \quad // \text{ Pkg: Prot}$ 

parse return as ( state, msg )
State[ctr] ← state
return msg

```

```

Send3(ctr, msg)

assert (State[ctr] ≠ ⊥)
state ← State[ctr]

return  $\xleftarrow{\text{invoke}} \text{Run3}(state, msg) \quad // \text{ Pkg: Prot}$ 

parse return as ( state, msg )
State[ctr] ← state
return msg

```

```

Send4(ctr, msg)

assert (State[ctr] ≠ ⊥)
state ← State[ctr]

return  $\xleftarrow{\text{invoke}} \text{Run4}(state, msg) \quad // \text{ Pkg: Prot}$ 

parse return as ( state, msg )
State[ctr] ← state
return msg

```

```

Send5(ctr, msg)

assert (State[ctr] ≠ ⊥)
state ← State[ctr]

return  $\xleftarrow{\text{invoke}} \text{Run5}(state, msg) \quad // \text{ Pkg: Prot}$ 

parse return as ( state, stop )
State[ctr] ← state
return stop

```

```

Reveal(ctr)

parse State[ctr] as  $\left( \begin{array}{c} U, u, V, ltk, acc, ni, nr, \\ kmac, sid, mess \end{array} \right)$ 

assert acc = true
assert RevTested[sid] = ⊥
RevTested[sid] ← false
 $k \xleftarrow{\text{invoke}} \text{Eval}(ltk, U, V, ni, nr, \text{true}) \quad // \text{ Pkg: Prf}$ 
return k

```

```

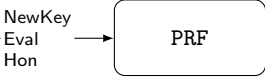
Test(ctr)

parse State[ctr] as  $\left( \begin{array}{c} U, u, V, ltk, acc, ni, nr, \\ kmac, sid, mess \end{array} \right)$ 

assert acc = true
assert Fresh[ctr]
assert RevTested[sid] = ⊥
RevTested[sid] ← true
if b then
     $k \xleftarrow{\$} \{0, 1\}^{256}$ 
else
     $k \xleftarrow{\text{invoke}} \text{Eval}(ltk, U, V, ni, nr, \text{true}) \quad // \text{ Pkg: Prf}$ 
return k

```

1.7 PRFideal Game

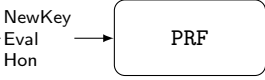


```
PRF
NewKey(ltk)
-----
kid ← (kid + 1)
if ltk = ⊥ then
    ltk  $\xleftarrow{id1} \{0, 1\}^{256}$ 
    LTK[kid] ← ltk
    H[kid] ← true
else
    LTK[kid] ← ltk
    H[kid] ← false
return kid
```

```
Eval(h, a, aa, c, d, e)
assert (LTK[h] ≠ ⊥)
if (H[h] = false ∨ ¬b) then
    k ← LTK[h]
    return prf(k, a, aa, c, d, e)
if PRF[( h, a, aa, c, d, e )] = ⊥ then
    temp  $\xleftarrow{id5} \{0, 1\}^{256}$ 
    PRF[( h, a, aa, c, d, e )] ← temp
    return temp
y ← PRF[( h, a, aa, c, d, e )]
return y
```

```
Hon(h)
assert (H[h] ≠ ⊥)
return H[h]
```

1.8 PRFreal Game



```
PRF
NewKey(ltk)
-----
kid ← (kid + 1)
if ltk = ⊥ then
    ltk  $\xleftarrow{id1}$  {0, 1}256
    LTK[kid] ← ltk
    H[kid] ← true
else
    LTK[kid] ← ltk
    H[kid] ← false
return kid
```

```
Eval(h, a, aa, c, d, e)
assert (LTK[h] ≠ ⊥)
if (H[h] = false ∨ ¬b) then
    k ← LTK[h]
    return prf(k, a, aa, c, d, e)
if PRF[( h, a, aa, c, d, e )] = ⊥ then
    temp  $\xleftarrow{id5}$  {0, 1}256
    PRF[( h, a, aa, c, d, e )] ← temp
    return temp
y ← PRF[( h, a, aa, c, d, e )]
return y
```

```
Hon(h)
assert (H[h] ≠ ⊥)
return H[h]
```




Figure 1: Reduction: $KX_NoPrf_RealPrf \approx KX_NoPrf_IdealPrf$ assuming prf

2 Advantages

Definition 1 (prf). For all adversaries \mathcal{A} , we define the asp-name advantage as

$$Adv(\mathcal{A}; PRF_{real}, PRF_{ideal}) := \left| \begin{array}{l} \Pr[\mathcal{A} \rightarrow PRF_{real} = 0] \\ - \Pr[\mathcal{A} \rightarrow PRF_{ideal} = 0] \end{array} \right|,$$

where PRF_{real} and PRF_{ideal} are defined in Sec. 1.8 and Sec. 1.7, respectively.

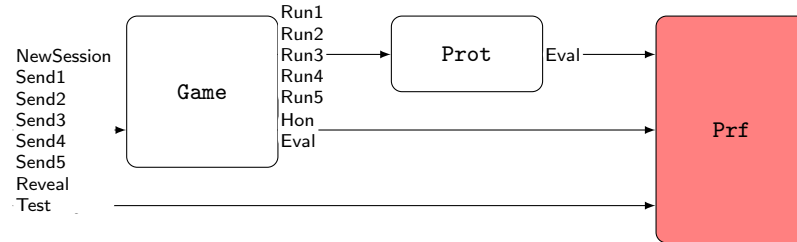
3 Gamehops

3.1 Equivalence between KX and KX_NoKey

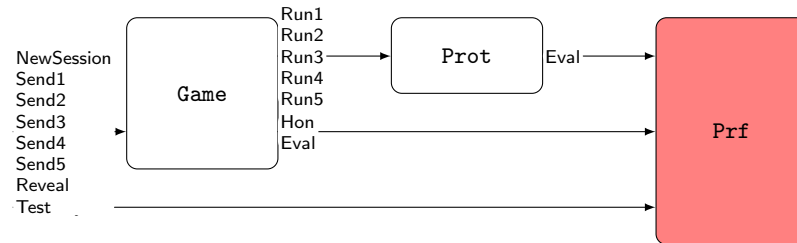
3.2 Equivalence between KX_NoKey and KX_NoPrf_RealPrf

3.3 Reduction to prf

3.3.1 Game $KX_NoPrf_RealPrf$ with Assumption Game PRF_{real} highlighted in red



3.3.2 Game $KX_NoPrf_IdealPrf$ with Assumption Game PRF_{ideal} highlighted in red



3.4 Equivalence between $RealKX_NoPrf_IdealPrf$ and $IdealKX_NoPrf_IdealPrf$