Gauge Hartwell

25 January, 2021

CS427

Homework 6

1. Libraries:

$\mathcal{L}_{\text{prf-real}}$

$k \leftarrow \{0,1\}^\lambda$

LOOKUP(m) :

return $F(k,m) \| F(k,F(k,m))$

$\mathcal{L}_{\text{prf-rand}}$

$T :=$ empty assoc. array

LOOKUP(m) :

if $T[m]$ undefined:

$T[m] \leftarrow \{0,1\}^{2n}$

return $T[m]$

Calling program:

F'

$m \leftarrow \{0,1\}^\lambda$

$z := Lookup(m)$

$a \| b := Lookup(z)$

if $b == F(k,z)$

return 1

return 0

Advantage:

Pr[F' ◊ prf-real == 1] == 1

Pr[F' ◊ prf-rand == $1/2^\lambda$

$1-1/2^\lambda$, non-negligible

2. Libraries

$\mathcal{L}_{\text{prf-real}}$

$k_1 \leftarrow \{0,1\}^\lambda$

$k_2 \leftarrow \{0,1\}^\lambda$

LOOKUP$(v_0 \| v_1)$:

$v_2 := F(k1, v_1) \oplus v_0$

$v_3 := F(k2, v_2) \oplus v_1$

return $v_2 \| v_3$

$\mathcal{L}_{\text{prf-rand}}$

$T := empty$

LOOKUP$(v_0 \| v_1)$:

if $T(v_0 \| v_1)$ is undefined:

$T(v_0 \| v_1) \leftarrow \{0,1\}^{2n}$

return $T(v_0 \| v_1)$

Calling Program:

```
CALL()
  x ← {0,1}^λ
  y ← {0,1}^λ
  a, b := Lookup(x, y)
  w, z := Lookup(a, y)
  if (x == w)
    return 1
  return 0
```

Advantage:

Pr[Call ◊ prf-real == 1] = 1

Pr[Call ◊ prf-rand == 1] == $1/2^{\lambda}$

$1 - 1/2^{\lambda}$, non-negligible

3. Starting function:

```
k ← {0,1}^λ

Lookup(r) :
  return G(F(k, r))
```

First, we will factor out the F function into a separate subroutine. This does not change how the library operates.

```
Lookup(r) :
  y := Lookup(r)
  return G(y)
```
◊
```
L_prf-real
k ← {0,1}^λ

Lookup(r) :
  return F(k, r)
```

Second, we will swap prf-real for prf-rand, which will have no effect on the operation of the library.

```
Lookup(r) :
  y := Lookup(r)
  return G(y)
```
◊
```
L_prf-rand
T := undefined

Lookup(r) :
  If (T[r] == undefined):
    T[r] ← {0,1}^λ
  return T[r]
```

Now, we can inline the subroutine, changing nothing about the operation of the library.

```
T := undefined

Lookup_F(r) :
  If (T[r] == undefined):
    T[r] ← {0,1}^λ
  y := T[r]
  return G(y)
```

Next, we can add T' and apply G to it. This means G is called once and stored in T'[r], instead of calling G every time the library is run. This changes nothing about how the library operates.

$T := undefined$
$T' := undefined$

$\underline{Lookup_F(r):}$
If (T[r] == undefined):
$T[r] \leftarrow \{0,1\}^\lambda$
$T'[r] := G(T[r])$
return $T'[r]$

Then, we can omit T because it is not serving any purpose anymore. This does not change how the library operates.

$T' := undefined$

$\underline{Lookup_F(r):}$
If (T'[r] == undefined):
$y \leftarrow \{0,1\}^\lambda$
$T'[r] := G(y)$
return $T'[r]$

Now we can factor out the lines within the if statement into a subroutine, which changes nothing about the operation of the library.

$T' := undefined$

$\underline{Lookup_F(r):}$
If (T'[r] == undefined):
$T'[r] := Lookup_G()$
return $T'[r]$

$\diamond$

$\mathcal{L}_{\text{prg-real}}$
$\underline{Lookup_G():}$
$s \leftarrow \{0,1\}^\lambda$
return $G(s)$

Through the security of prg, we can swap prg-real for prg-rand, which changes nothing about the operation of the library.

$T' := undefined$

$\underline{Lookup_F(r):}$
If (T'[r] == undefined):
$T'[r] := Lookup_G()$
return $T'[r]$

$\diamond$

$\mathcal{L}_{\text{prg-rand}}$
$\underline{Lookup_G():}$
$r \leftarrow \{0,1\}^{\lambda+\ell}$
return r

Finally, we can inline the prg-rand subroutine, and end up with our final function, which is

indistinguishable from our starting function.