Security of OTP (Secret Sharing)

HW1 will be released today

BIGPICTURE OF SECURITY DEFS

- Define 2 libraries with a common interface but different internals
- Adv is a calling program (trying to behave differently in presence of 2 libraries)
- If no Adv behaves differently then "interface leaks no information about internal differences"

Ex:

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

```
ctxt(m):
    k ← {0,1}^λ
    ret k ⊕ m
```

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

```
ctxt(m):
    c ← {0,1}^λ
    ret c
```

Claim: (last time)

for all A, $\Pr[A \diamond \mathcal{L}_{otp\text{-}real} \Rightarrow 1] = \Pr[A \diamond \mathcal{L}_{otp\text{-}rand} \Rightarrow 1]$

($A \diamond \mathcal{L}_{otp\text{-}real}$: A linked to the library; $A \diamond \mathcal{L}_{otp\text{-}rand} \Rightarrow 1$: event that linked program outputs 1)

(shorthand: $\mathcal{L}_{otp\text{-}real} \equiv \mathcal{L}_{otp\text{-}rand}$)

they are interchangeable

Notes:

- Adv can choose m
- Adv can "know everything" about 2 libraries
- only thing Adv doesn't "know" is
  - which $\mathcal{L}$ he is currently linked to
  - values of privately scoped variables
- Kerckhoff's principle (1883): Assume Adv knows everything about your system except the key

Q: What happens if Adv sees 2 ctxts under same k?

$$\boxed{\begin{array}{l}\underline{ctxt(m_1, m_2):} \\ k \leftarrow \{0,1\}^\lambda \\ \text{ret } k \oplus m_1,\ k \oplus m_2\end{array}} \stackrel{?}{\equiv} \boxed{\begin{array}{l}\underline{ctxt(m_1, m_2):} \\ c_1 \leftarrow \{0,1\}^\lambda \\ c_2 \leftarrow \{0,1\}^\lambda \\ \text{ret } c_1, c_2\end{array}}$$

Claim: $\not\equiv$, need to show A that behaves differently when linked to left/right $\mathcal{L}$

A:

$(c_1, c_2) \leftarrow ctxt(0^\lambda, 1^\lambda)$

// if in "left world", $c_1 = k$, $c_2 = k \oplus 1^\lambda$

return $c_1 \oplus c_2 \stackrel{?}{=} 1^\lambda$

When $A \diamond \mathcal{L}_{left}$: $c_1 = k \oplus 0^\lambda = k$

$c_2 = k \oplus 1^\lambda$

$\Rightarrow c_1 \oplus c_2 = 1^\lambda \Rightarrow$ A returns true, always

When $A \circ \mathcal{L}_{right}$: c_1, c_2 uniformly chosen

$$\Pr[c_1 \oplus c_2 = 1^\lambda]$$
$$= \Pr[c_2 = 1^\lambda \oplus c_1]$$

c_2: fixed at time of breakpoint

$1^\lambda \oplus c_1$: about to be chosen uniformly

$$= 1/2^\lambda \neq 1$$

BIG PICTURE of breaking security:

- write down code of Adv (calling program)
- show that $\Pr[A \circ \mathcal{L}_{left} \Rightarrow 1] \neq \Pr[A \circ \mathcal{L}_{right} \Rightarrow 1]$

Better definition:

(prev def for OTP is very specific to OTP)

Def: enc. scheme Σ has one-time secrecy if

$\Sigma = (KeyGen, Enc, Dec)$

$\mathcal{L}^{\Sigma}_{ots\text{-}L}$	$\equiv$	$\mathcal{L}^{\Sigma}_{ots\text{-}R}$
Query (m_L, m_R):		Query (m_L, m_R):
$k \leftarrow \Sigma.KeyGen$		$k \leftarrow \Sigma.KeyGen$
ret $\Sigma.Enc(k, m_L)$		ret $\Sigma.Enc(k, m_R)$

only diff between $\mathcal{L}$'s is choice of ptxt

$\Rightarrow$ "seeing 1 ctxt leaks no info about choice of ptxt"

Claim: OTP has one-time secrecy [see PDF]

BIG PICTURE of proving security:

- proving security means showing $\mathcal{L}_1 \equiv \mathcal{L}_2$
- start w/ $\mathcal{L}_1$ and make sequence of small modifications
- justify why each modification has no effect to calling program
- end at $\mathcal{L}_2$