Also possible to use RSA to build PKE:

"Textbook RSA" (How NOT to encrypt): Consider the following candidate of a PKE scheme from RSA:
- Setup (1^2): Sample (n,e,d) where N=pp and ed = 1 (mod φ(N)). Output pk = (N,e) and sk = (N,d)
- Encrypt (pk, m): Output c ← m^e \{ Correct since c^d = (m^e)^d = m^1 = m (mod N)
- Decrypt (sk, c): Output m ← c^d

Correctness follows from correctness of TDP.

How about security? NO...

1. Security of TDP says that inserting random element should be difficult
   - Does not apply if messages chosen adversarially (e.g., semantic security definition)
   - Does not say anything about hiding preimage (e.g., F(pp,x) can leak information about x so long
     as leakage is not sufficient to fully recover x — this is a weaker property than full indistinguishability)

   a. This scheme is deterministic: cannot be semantically secure!

   NEVER use textbook RSA!

To use RSA/TPPs to construct a PKE scheme, we will use a similar strategy as in the FD1 signature construction:
- Setup (1^2): Sample (pp,td) ← Setup (1^2) for the TDP scheme and output pk = pp and sk = td
- Encrypt (pk, m): Sample \( x \in \mathbb{Z}_n^* \) from domain of TDP
  \[ \text{Let } k \leftarrow H(x) \text{ where } H: X \rightarrow K_0 \text{ is an (ideal) hash function and } K_0 \text{ is the key space for an} \]
  \[ \text{symmetric authenticated encryption scheme} \]
  \[ \text{Compute } y \leftarrow F(pp, x) \text{ and } ct \leftarrow \text{Enc}_{AE}(k, m) \]
  \[ \text{Output } (y, ct) \]
- Decrypt (sk, ct' = (y, ct)):

This is an example of hybrid encryption or KEM: y is used to encapsulate the key and ct' is an encryption under k.

**Theorem.** If \( F \) is a trapdoor permutation and \( H \) is modeled as a random oracle, then the above encryption scheme is semantically secure. [In fact, this scheme is CCA-secure in the random oracle model]

**Proof intuition.** Given a ciphertext \((y, ct')\) and public key \(pk = pp\):
- Adversary cannot compute \( x \) from \( y \) (by security of TDP — once \( x \) is uniform)
- Adversary cannot evaluate \( H \) on \( x \), so \( k \) is uniformly random and hidden from adversary
- Semantic security follows from semantic security of symmetric encryption scheme

**RSA instance:**
- Setup (1^2): Sample (n,e,d) where N=pp and ed = 1 (mod φ(N)). Output pk = (N,e) and sk = (N,d)
- Encrypt (pk, m):
  \[ \text{Sample } x \in \mathbb{Z}_n^* \text{ and compute } y \leftarrow x^e \text{ (mod N).} \]
  \[ \text{Output } (y, ct) \]
  \[ \text{Compute } k \leftarrow H(x) \text{ and compute } ct' \leftarrow \text{Enc}_{AE}(k, m). \]
- Decrypt (sk, ct):
  \[ \text{Compute } x \leftarrow y^d \text{ (mod N), } k \leftarrow H(k), \text{ and output } m \leftarrow \text{Dec}_{AE}(k, ct') \]
In practice: Most widely-used standard for RSA encryption is PKCS1 (by RSA Labs)

\[ \Rightarrow \text{Has shorter ciphertexts if we are encrypting a single } Z_n \text{ element (no need for KEM + symmetric component)} \]

General approach: Suppose \( N \) is 2048 bits and we want to encrypt 256-bit messages

\[ \Rightarrow \text{we will first apply a randomized padding to } m \text{ to obtain a 2048-bit padded message} \]

PKCS1 padding:

\[
\begin{array}{c}
\text{00 02 non-zero random bytes 00 } m \\
\end{array}
\]

16 bits 5 bits where \( s \ll t \)

Encryption: Compute \( m_{\text{pad}} \leftarrow \text{PKCS}(m) \) and set \( c \leftarrow m_{\text{pad}}^e \) \[ \text{[i.e., directly apply RSA trapdoor permutation to padded]} \]

Decryption: Compute \( m_{\text{pad}} \leftarrow c^d \) and recover \( m \) from \( m_{\text{pad}} \)

In SSL v3.0: during the handshake, server decrypts client's message and checks if resulting \( m_{\text{pad}} \) is well-formed

\[ \Rightarrow \text{scheme is vulnerable to a chosen-ciphertext attack!} \]

\[ \Rightarrow \text{allows adversary to eavesdrop on connection} \]

Devastating attack on SSL v3.0 and very hard to fix: need to change both servers + clients!

\[ \Rightarrow \text{tl;dr: fix is to set } m \in \mathbb{Z}_N^* \text{ if decryption ever fails and proceed normally (never alert client if padding is malformed) - setup fails at a later point in time, but hopefully no critical information is leaked...} \]

Take-away: PKCS1 is not CCA-secure which is very problematic for key exchange

\[ \Rightarrow \text{Absence of security proof should always be troubling...} \]

New standard: Optimal Asymmetric Encryption Padding (OAEP) [1994]

\[ \Rightarrow \text{Can be shown to be CCA-secure in random oracle model} \]

\[ \Rightarrow \text{version 2.0} \]