Signatures from trapdoor permutations (the full domain hash):

In order to appeal to security of TDP, we need that the argument to $F^{-1}(td, \cdot)$ to be <u>random</u>

<u>Idea</u>: hash the message first and sign the hash value (often called "hash-and-sign")

↳ <u>Another benefit</u>: Allows signing long messages (much larger than domain size of TDF)

<u>FDH construction</u>:

- $\text{Setup}(1^\lambda)$: Sample $(pp, td) \leftarrow \text{Setup}(1^\lambda)$ for the TDP and output $vk = pp$, $sk = td$
- $\text{Sign}(sk, m)$: Output $\sigma \leftarrow F^{-1}(td, H(m))$
- $\text{Verify}(vk, m, \sigma)$: Output 1 if $F(pp, \sigma) = H(m)$ and 0 otherwise

<u>Theorem</u>. If F is a trapdoor permutation and H is modeled as an <u>ideal hash function</u> (i.e., random oracle), then the full domain hash signature scheme defined above is secure.

↳ function that behaves like a truly random function (does not exist but functions like SHA-256 or SHA-3 believed to be "close enough")

<u>Proof Intuition</u>: To forge a signature on a message m, need to compute $F^{-1}(td, H(m))$ where $H(m)$ is <u>uniformly random</u> in domain of the permutation — security follows from TDP.

<u>Caveat</u>: still need to simulate signing queries ⇝ relies on "programming" the random oracle (see office hours)

<u>Recap</u>: RSA-FDH signatures:

$\text{Setup}(1^\lambda)$: Sample modulus N, e, d such that $ed = 1 \pmod{\varphi(N)}$ — typically $e = 3$ or $e = 65537$

(Some (partial) attacks can exploit very small public exponent ($e=3$))

Output $vk = (N, e)$ and $sk = (N, d)$

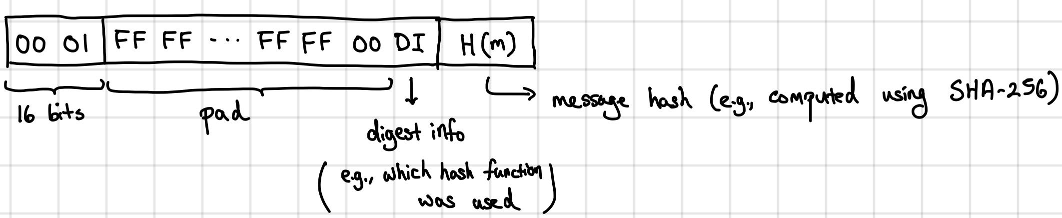
$\text{Sign}(sk, m)$: $\sigma \leftarrow H(m)^d$ [Here, we are assuming that H maps into $\mathbb{Z}_N^*$]

$\text{Verify}(vk, m, \sigma)$: output 1 if $H(m) = \sigma^e$ and 0 otherwise

<u>Standard</u>: PKCS1 v1.5 (typically used for signing certificates)

↳ Standard cryptographic hash functions hash into a 256-bit space (e.g., SHA-256), but FDH requires <u>full domain</u>

↳ PKCS 1 v1.5 is a way to <u>pad</u> hashed message before signing:



↳ Padding important to protect against chosen message attacks (e.g., preprocess to find messages m_1, m_2, m_3 where $H(m_1) = H(m_2) \cdot H(m_3)$)
(but this is <u>not</u> a full-domain hash and <u>cannot</u> prove security under RSA — can make stronger assumption...)

Also possible to use RSA to build PKE:

"Textbook RSA" (How NOT to encrypt): Consider the following candidate of a PKE scheme from RSA:

- $\mathrm{Setup}(1^\lambda)$: Sample (N, e, d) where $N = pq$ and $ed = 1 \pmod{\varphi(N)}$. Output $pk = (N, e)$ and $sk = (N, d)$
- $\mathrm{Encrypt}(pk, m)$: Output $c \leftarrow m^e$
- $\mathrm{Decrypt}(sk, ct)$: Output $m \leftarrow c^d$

} Correct since $c^d = (m^e)^d = m^{ed} = m^1 = m \pmod N$

Correctness follows from correctness of TDP.

How about security? NO.

1. Security of TDP says that inverting 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)
2. This scheme is deterministic: cannot be semantically secure!

NEVER use textbook RSA!

To use RSA / TDPs to construct a PKE scheme, we will use a similar strategy as in the FDH signature construction:

- $\mathrm{Setup}(1^\lambda)$: Sample $(pp, td) \leftarrow \mathrm{Setup}(1^\lambda)$ for the TDP scheme and output $pk = pp$ and $sk = td$
- $\mathrm{Encrypt}(pk, m)$: Sample $x \xleftarrow{R} \mathcal{X}$ from domain of TDP — Scheme is randomized!

  Let $k \leftarrow H(x)$ where $H: \mathcal{X} \to \mathcal{K}$ is an (ideal) hash function and $\mathcal{K}$ is the key-space for an symmetric authenticated encryption scheme

  Compute $y \leftarrow F(pp, x)$ and $ct' \leftarrow \mathrm{Enc}_{AE}(k, m)$

  Output (y, ct')
- $\mathrm{Decrypt}(sk, ct' = (y, ct))$: Compute $x \leftarrow F^{-1}(td, y)$, $k \leftarrow H(x)$, and output $m \leftarrow \mathrm{Dec}_{AE}(k, ct')$

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 an ideal hash function, 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 — since x is uniform)
- Adversary cannot evaluate H on x, so k is uniformly random and hidden from adversary (if H is ideal...)
- Semantic security follows from semantic security of symmetric encryption scheme.

RSA instantiation:

- $\mathrm{Setup}(1^\lambda)$: Sample (N, e, d) where $N = pq$ and $ed = 1 \pmod{\varphi(N)}$. Output $pk = (N, e)$, $sk = (N, d)$
- $\mathrm{Encrypt}(pk, m)$: Sample $x \xleftarrow{R} \mathbb{Z}_N^*$ and compute $y \leftarrow x^e \pmod N$.

  Compute $k \leftarrow H(x)$ and compute $ct' \leftarrow \mathrm{Enc}_{AE}(k, m)$.

  } Output (y, ct')
- $\mathrm{Decrypt}(sk, ct)$: Compute $x \leftarrow y^d \pmod N$, $k \leftarrow H(k)$, and output $m \leftarrow \mathrm{Dec}_{AE}(k, ct')$.

In practice: Most widely-used standard for RSA encryption is PKCS1 (by RSA labs)

↳ Has shorter ciphertexts if we are encrypting a single $\mathbb{Z}_N$ element (no need for KEM + symmetric component)

(helpful if PKE just used to encrypt short token or metadata)

General approach: suppose N is 2048 bits and we want to encrypt 256-bit messages

↳ we will first apply a randomized padding to m to obtain a 2048-bit padded message

PKCS 1 padding:
(mode 2)

Encryption: Compute $m_{pad} \leftarrow \text{PKCS}(m)$ and set $c \leftarrow m_{pad}^e$

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

[i.e., directly apply RSA trapdoor permutation to padded message]

In SSL v3.0: during the handshake, server decrypts client's message and checks if resulting m_{pad} is well-formed (i.e., has valid PKCS1 padding) and rejects if not

↳ scheme is vulnerable to a chosen-ciphertext attack!

↳ allows adversary to eavesdrop on connection

Devastating attack on SSL 3.0 and very hard to fix: need to change both servers + clients!

↳ TLS 1.0: fix is to set $m \xleftarrow{R} \mathbb{Z}_N^*$ 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

↳ Absence of security proof should always be troubling...

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

↳ Can be shown to be CCA-secure in random oracle model

} standardized in PKCS1 version 2.0