RSA and Digital Signatures

Schedule for today

Recap

More on RSA
1. Making RSA IND-CPA/IND-CCA secure
2. Hybrid encryption

Digital Signatures
1. What are Digital Signatures?
2. Formalizing security
3. Simple RSA signatures and why they are not secure
4. The RSA-FDH signature scheme
5. Proving RSA-FDH secure

What we did last time

Public Key Encryption

True or false?

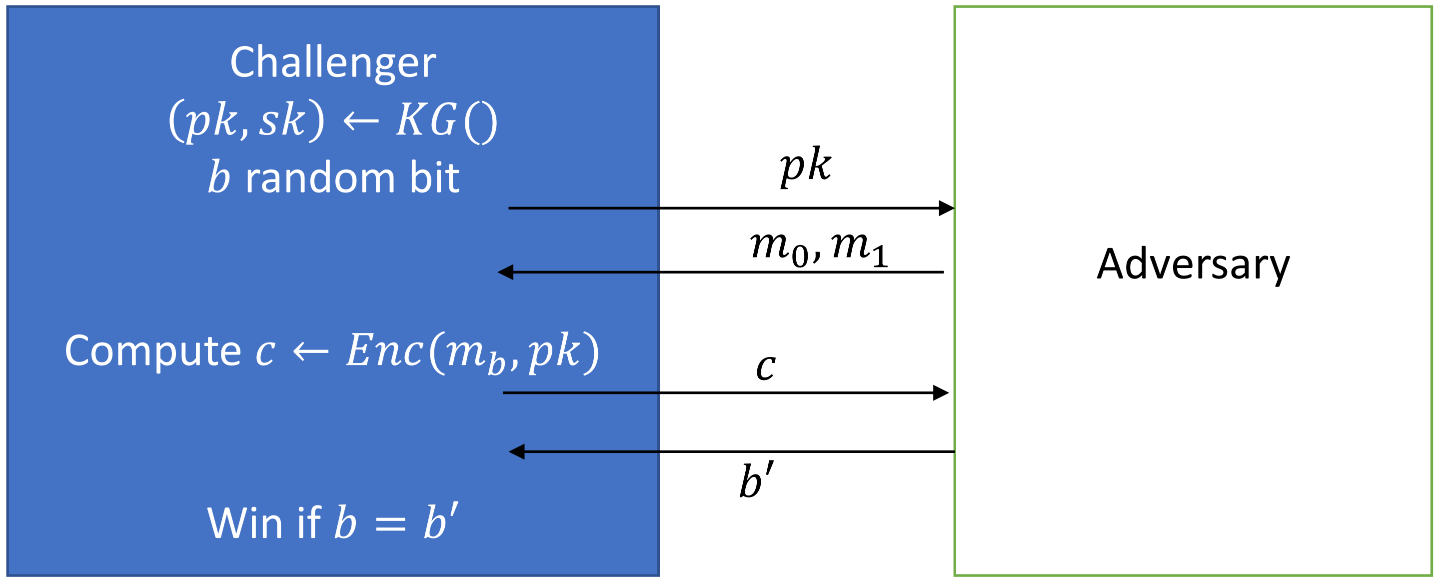
Public Key Encryption allows any two parties to confidentially communicate with each other, without knowing anything about each other.

Security of PKE

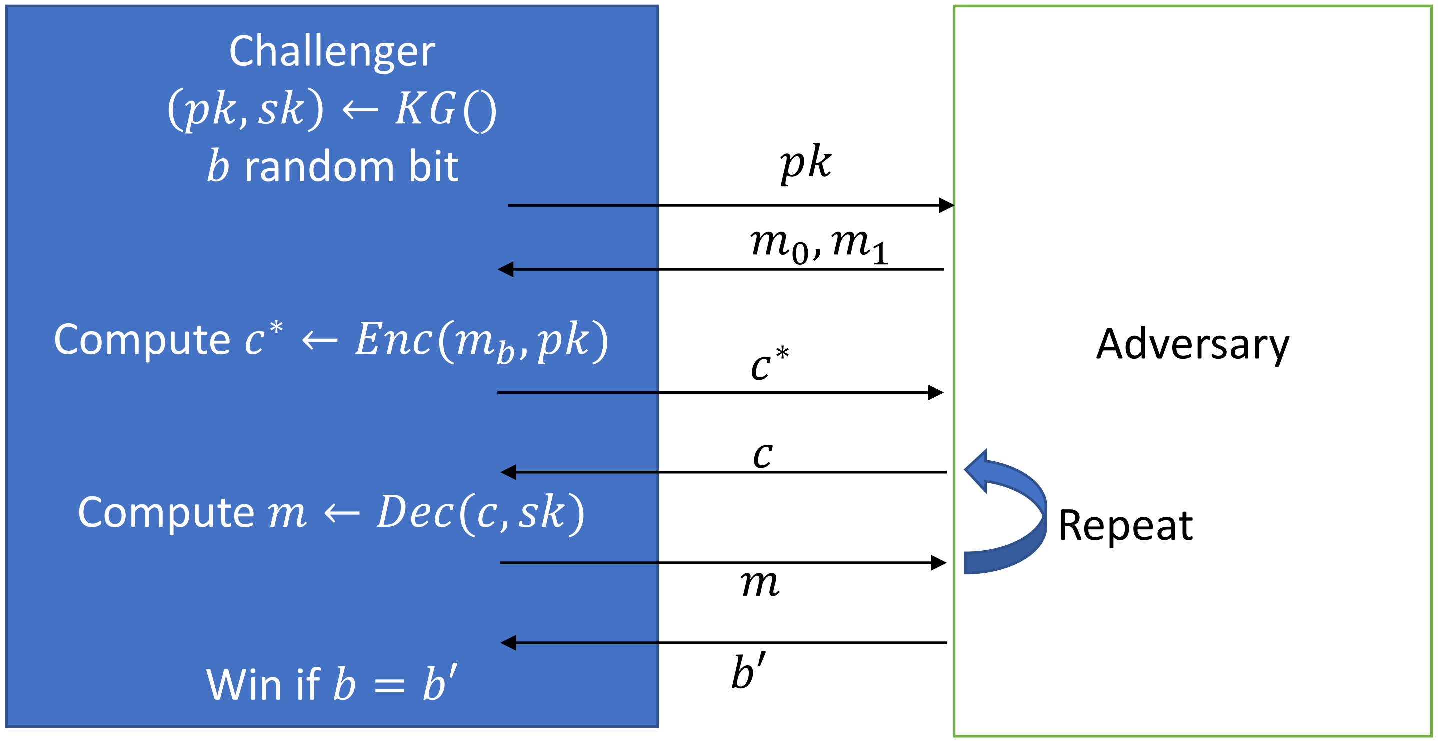
True or False?

IND-CPA is <u>the strongest security notion</u> for Public Key encryption schemes we know of.

Defining security of PKE – IND-CPA



Defining security of PKE – IND-CCA



RSA

Which statement/statements is/are true?

RSA is secure if

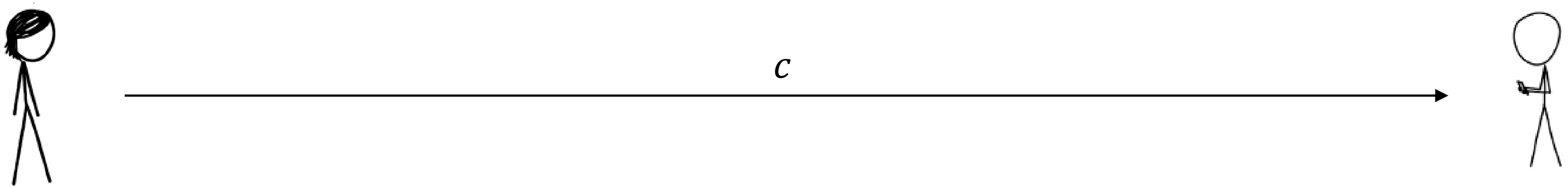
1. It is hard to compute e'th roots modulo a biprime N.
2. It is hard to compute e'th powers modulo a biprime N.
3. It is hard to factor a number N into its prime factors.

More about RSA

The RSA cryptosystem

Key Generation

1. Find two large primes p, q and e with $\gcd\big(e, (p-1)\cdot(q-1)\big) = 1$
2. Compute $N = p \cdot q$
3. Find d such that $d \cdot e = 1 \; mod \; (p-1)(q-1)$

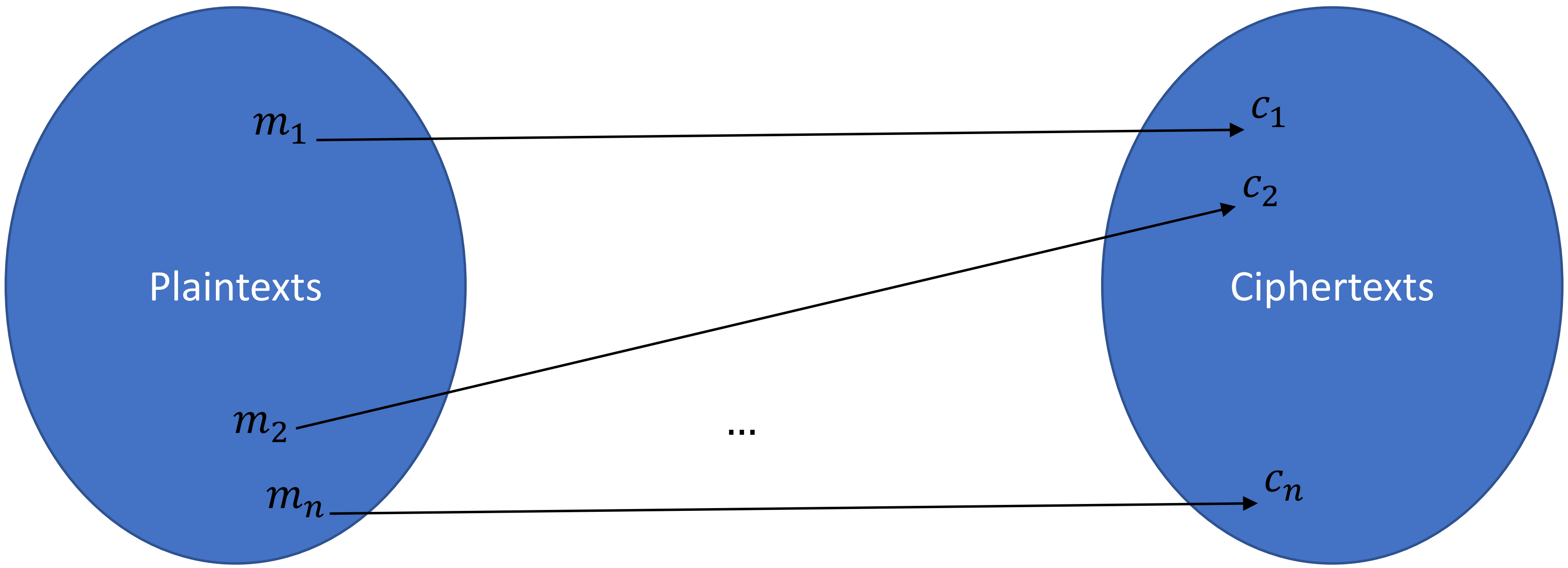


Compute $c = m^e \; mod \; N$

Not IND-CPA/IND-CCA secure

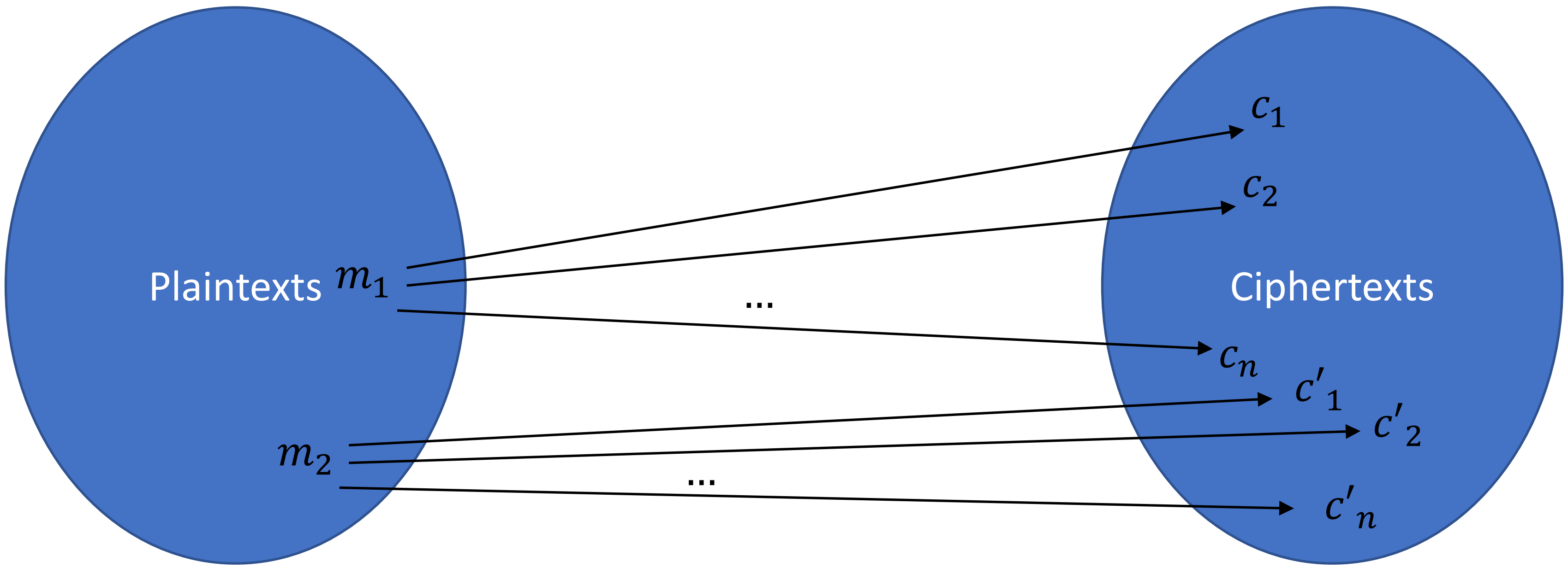
Compute $m = c^d \; mod \; N$

Core of the problem for IND-CPA: no randomness

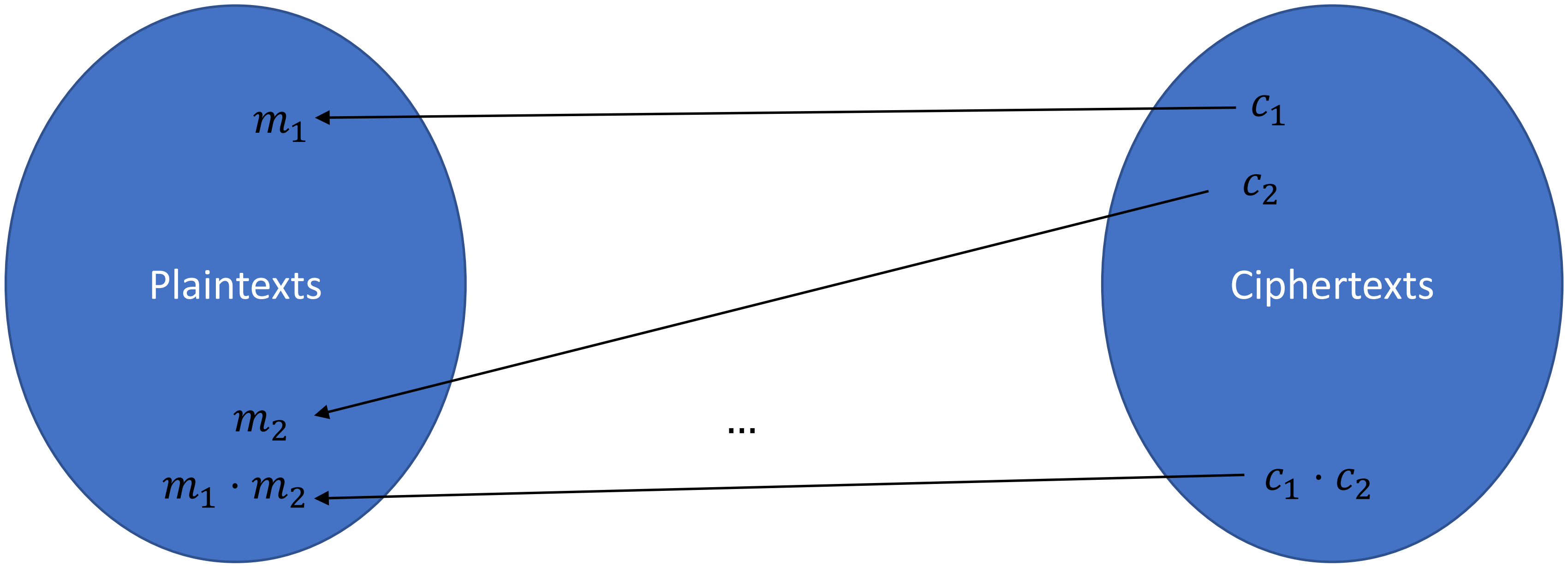


As many messages as ciphertexts, encryption/decryption is bijection

Solving the IND-CPA problem

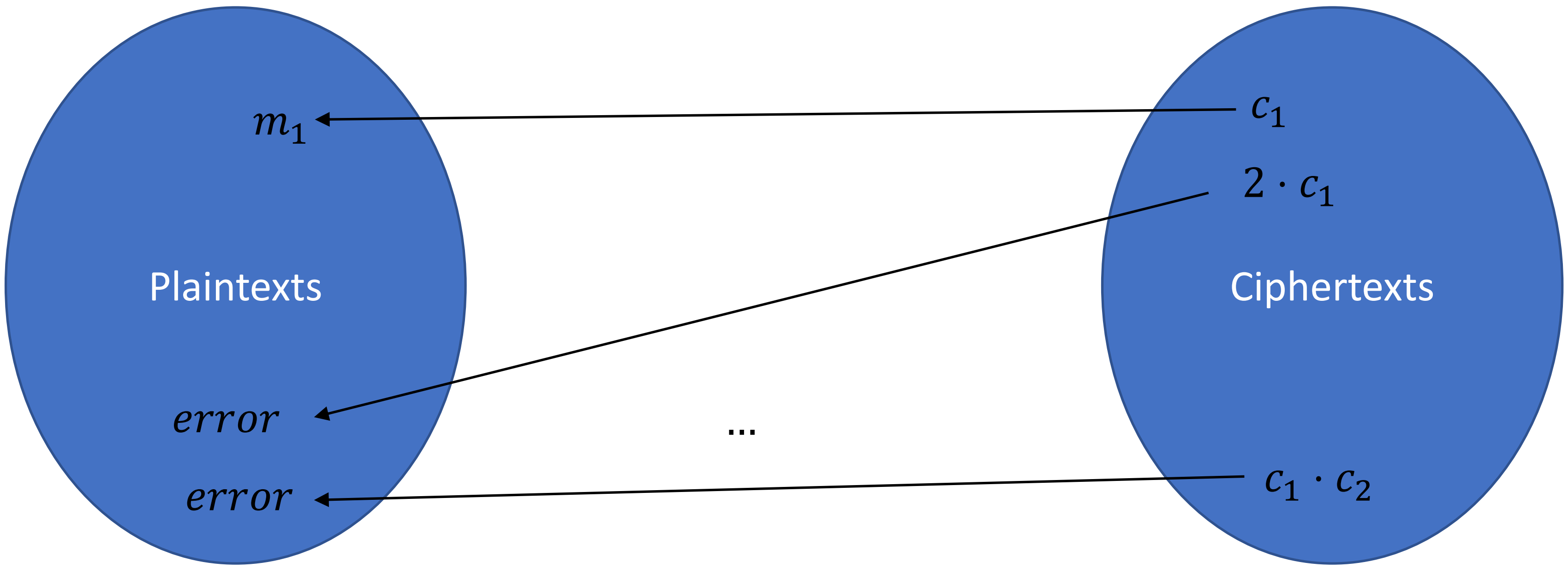


Core of problem for IND-CCA: Malleable ciphertexts



In RSA we can change ciphertexts such that their plaintexts change in a predictable way

Solving the IND-CCA problem



RSA-OAEP

Let $k = 8 \cdot \lfloor \log_8 N \rfloor$ and $k_0, k_1 > 128, n = k - k_0 - k_1$

Messages $m \in \{0,1\}^n$
Hash functions G: $\{0,1\}^{k_0} \rightarrow \{0,1\}^{n+k_1}$, $H: \{0,1\}^{n+k_1} \rightarrow \{0,1\}^{k_0}$

Encryption for m, N, e:

1. Sample random bit string $R \in \{0,1\}^{k_0}$
2. Compute $A = [\left(m \middle| 0^{k_1}\right) \oplus G(R), R \oplus H\left(\left(m \middle| 0^{k_1}\right) \oplus G(R)\right)]$
3. Set $c = A^e \ mod\ N$

Decrypt RSA-OAEP

Messages $m \in \{0,1\}^n$
Hash functions $G: \{0,1\}^{k_0} \rightarrow \{0,1\}^{n+k_1}$, $H: \{0,1\}^{n+k_1} \rightarrow \{0,1\}^{k_0}$

Format: $[(m|0^{k_1}) \oplus G(R), R \oplus H\left((m|0^{k_1}) \oplus G(R)\right)]$

<u>Decryption for c, N, d:</u>

1. Compute $A' = c^d \ mod \ N$ and check if $A' < 2^k$
2. Let $A' = [B_0, B_1]$ and compute $R' = H(B_0) \oplus B_1$
3. Compute $m' = B_0 \oplus G(R')$. If m' ends with k_1 0s then recover m

Why RSA-OAEP works

Messages $m \in \{0,1\}^n$

Hash functions G: $\{0,1\}^{k_0} \rightarrow \{0,1\}^{n+k_1}$, $H: \{0,1\}^{n+k_1} \rightarrow \{0,1\}^{k_0}$

Format: $\mathrm{A} = [\left(m \middle| 0^{k_1}\right) \oplus G(R), R \oplus H\left(\left(m \middle| 0^{k_1}\right) \oplus G(R)\right)]$

IND-CPA: every choice of R, m gives different A (2^{k_0} many)

Why RSA-OAEP works

Messages $m \in \{0,1\}^n$

Hash functions $G: \{0,1\}^{k_0} \rightarrow \{0,1\}^{n+k_1}$, $H: \{0,1\}^{n+k_1} \rightarrow \{0,1\}^{k_0}$

Format: $\mathrm{A} = [(m|0^{k_1}) \oplus G(R), R \oplus H\left((m|0^{k_1}) \oplus G(R)\right)]$

Changing 1 bit in m or R creates entirely different block

Related messages don't have an algebraic relation!

Hybrid encryption

AES vs. RSA on a modern AMD Ryzen 9 5950X from 2020. 16 cores but we only use 1.

AES-128

1. It takes around 16 cycles per byte (https://bench.cr.yp.to/results-stream.html) to encrypt/decrypt AES-128. For a whole ciphertext of 16 bytes, that is around 256 cycles.
2. The processor runs at 3.400 MHz, i.e. it performs 3.400.000.000 cycles per second.
3. One core can approximately encrypt/decrypt 13.300.000 ciphertexts per second.

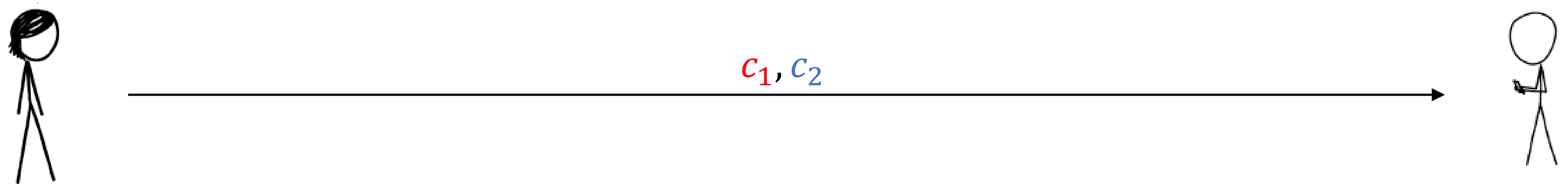
RSA w/ 2048 bit keys

1. Encryption with small exponent: ≈ 12.000 cycles
2. Decryption: $\approx 2.400.000$ cycles

(from https://bench.cr.yp.to/results-kem.html)

Hybrid encryption (key encapsulation)

Use PKE scheme Enc^{Pub}, Dec^{Pub}
together with SKE scheme Enc^{Sym}, Dec^{Sym}

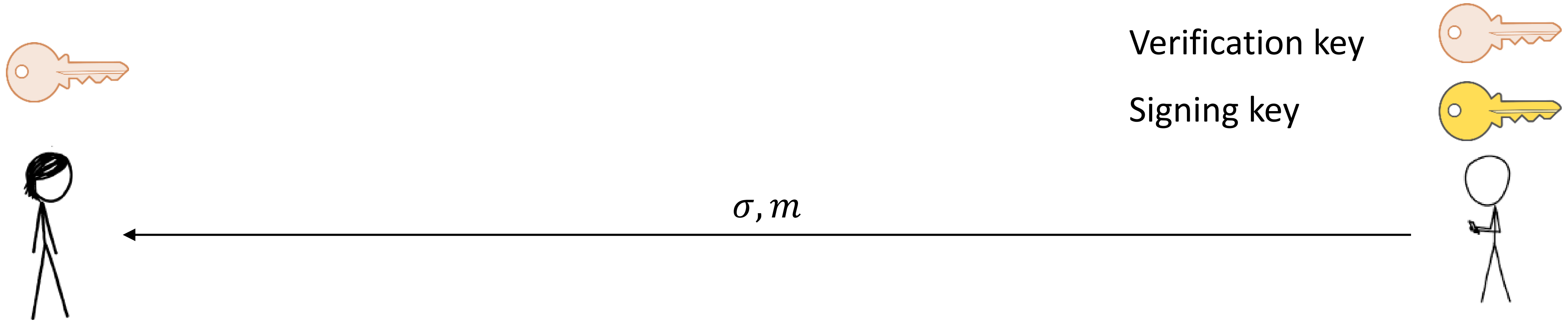


1. Choose symmetric key k.
2. Compute $c_1 \leftarrow Enc^{Pub}(k, pk)$.
3. Compute $c_2 \leftarrow Enc^{Sym}(m, k)$.

1. Recover $k \leftarrow Dec^{Pub}(c_1, sk)$.
2. Decrypt $\mathrm{m} \leftarrow Dec^{Sym}(c_2, k)$.

Digital Signatures

Digital Signatures



Verify m, σ
using vk

Sign m as
σ using sk

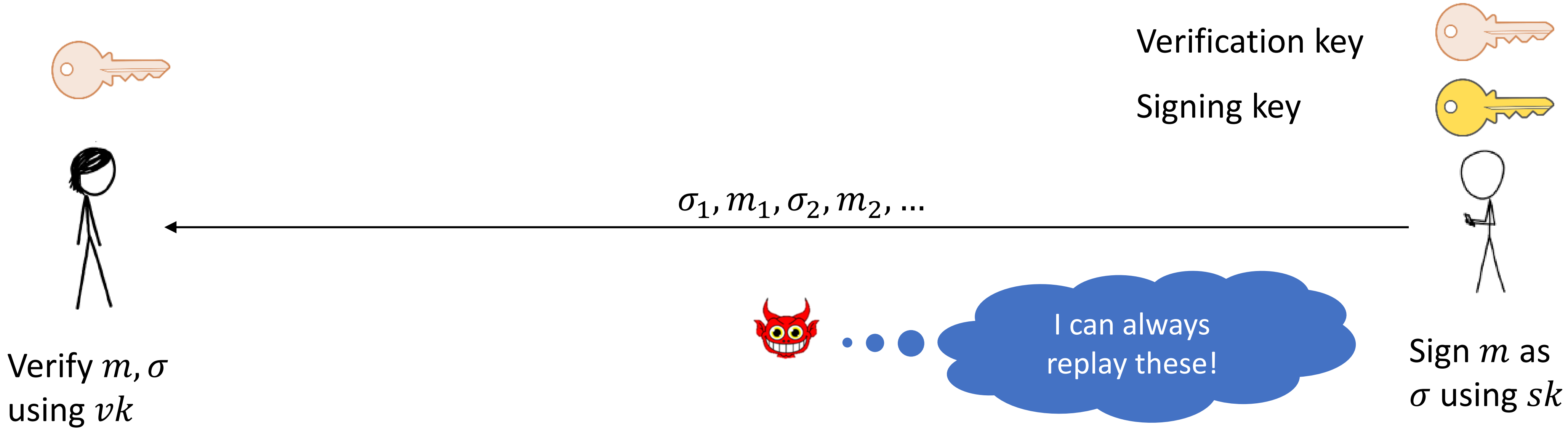
Algorithms:
$(vk, sk) \leftarrow KG$
$\sigma \leftarrow Sign(m, sk)$
$\{\perp, \top\} \leftarrow Verify(m, \sigma, vk)$

Use cases of digital signatures

- ``digital'' equivalent of signing a contract (NemID/MitID)
- Building authenticated channels over insecure network
- Software integrity
- Transactions in cryptocurrencies

Defining Security

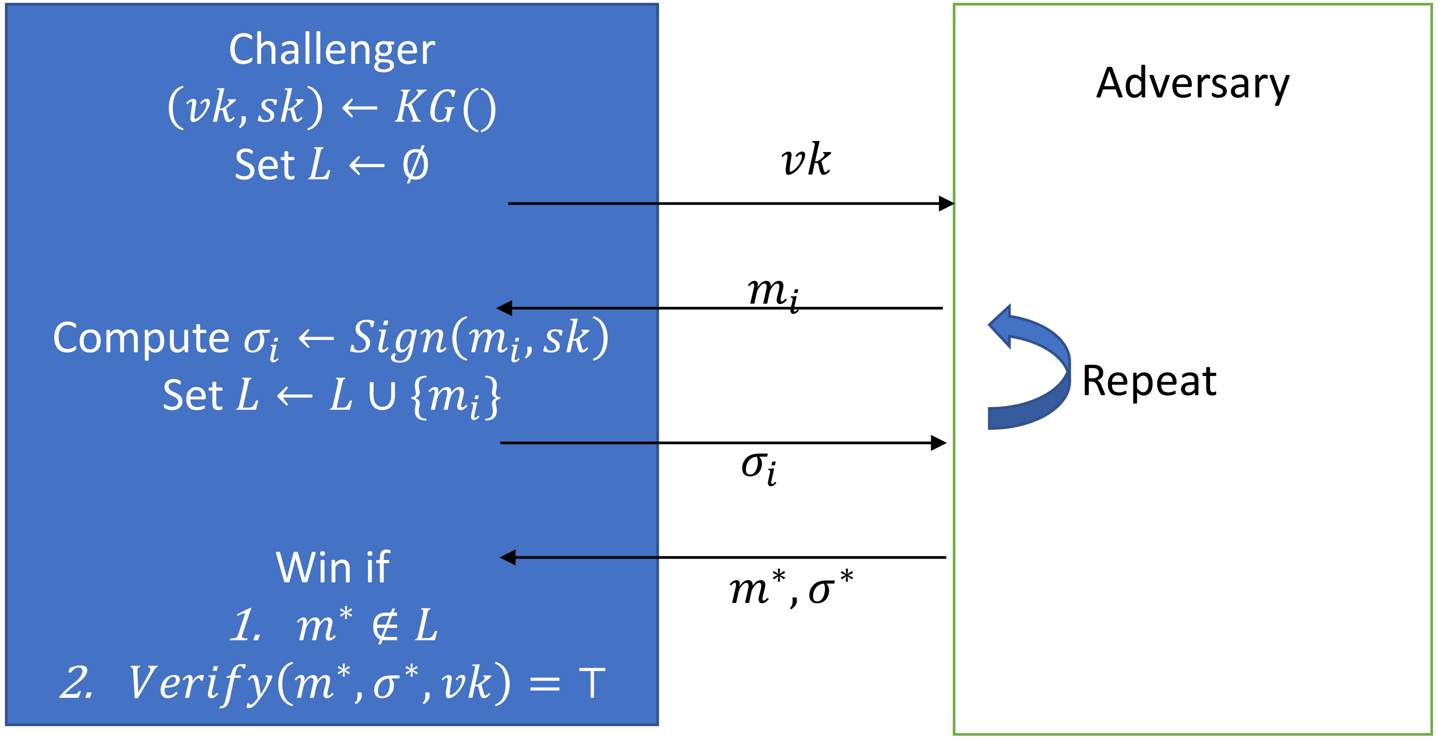
MACs for public key setting!



Unforgeability:

No adversary with vk and message/signature pairs $m_1, \sigma_1, \ldots$ should be able to make new m, σ

EUF-CMA for Signatures



Signatures from RSA: the wrong way

Signing key: secret d

Verification key: N, e

Message m $\xrightarrow{\text{Encrypt with private key}}$ $\sigma = m^d \ mod \ N$

Verify m, σ:

Check that $m = \sigma^e \ mod \ N$

This clearly fails!

Counterexample 1

Generate signature on ``random'' message:

1. Let $\mathrm{pk} = (N, e)$
2. Fix a random element $\sigma \in Z_N^*$
3. Compute $m = \sigma^e \ mod \ N$

(m, σ) is valid by construction

Counterexample 2 – Inspired by Homework 2

We want to forge a signature on m

1. Choose $m_1 \in Z_N^*$, compute $m_2 \leftarrow \frac{m}{m_1} mod\ N$

2. Ask EUF-CMA oracle to compute $\sigma_1 \leftarrow Sign(m_1, sk), \sigma_2 \leftarrow Sign(m_2, sk)$

3. Then $\sigma = \sigma_1 \cdot \sigma_2 = m_1^d \cdot m_2^d = m^d$ is a valid signature on m!

Digital Signatures using RSA: RSA-FDH

Signing key: secret d

Verification key N, e

Cryptographic hash $H: \{0,1\}^* \to Z_N^*$

$$\text{Message } m \xrightarrow{\text{Hash } H} h = H(m) \xrightarrow{\text{Encrypt with private key}} \sigma = h^d \bmod N$$

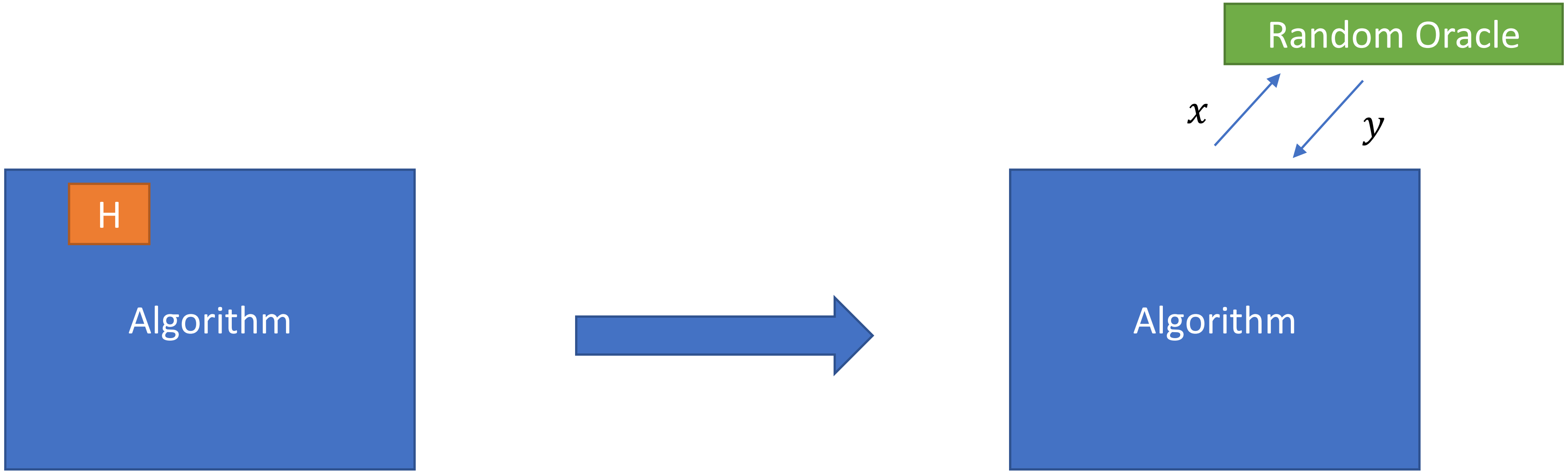
Verify m, σ:

Check that $H(m) = \sigma^e \bmod N$

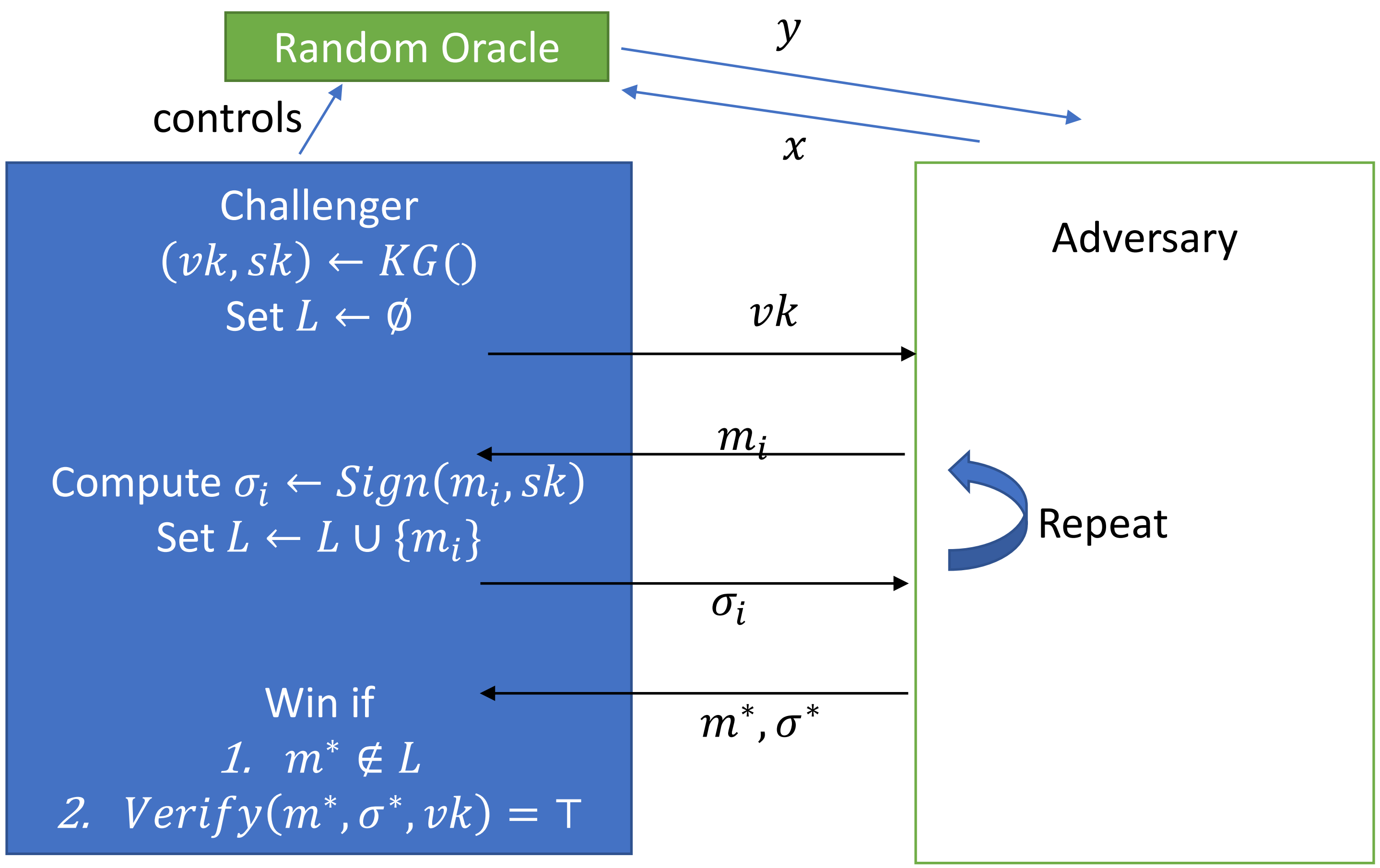
Any RSA instance for encryption can also be used for signing!

EUF-CMA security

Recap from Problem Sheet 5: the Random Oracle Model

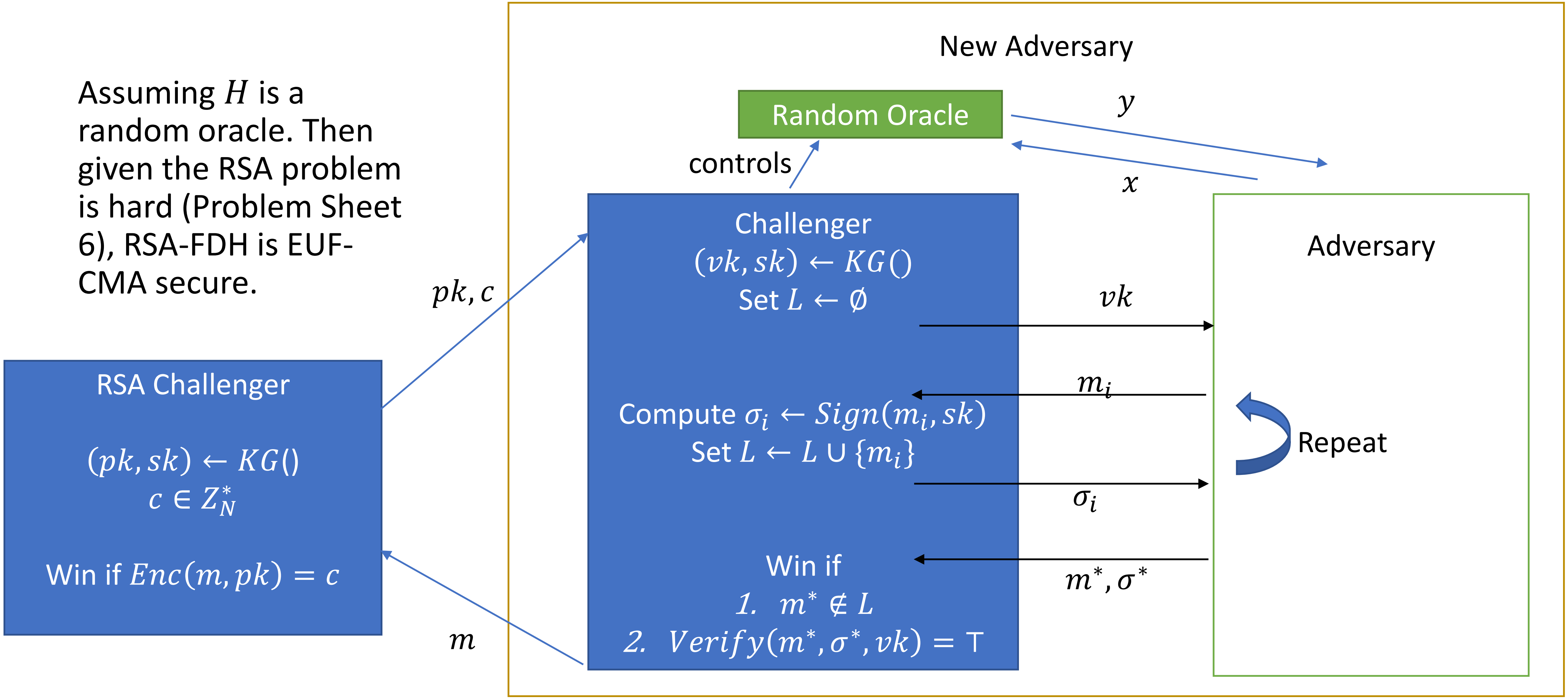


Looking at EUF-CMA



What we prove

Assuming H is a random oracle. Then given the RSA problem is hard (Problem Sheet 6), RSA-FDH is EUF-CMA secure.



Summary

When using RSA, use RSA-OAEP to make it IND-CCA secure (16.2.1 in the book)

Hybrid encryption for long messages

Digital signatures

RSA signatures using RSA-FDH