Hybrid Encryption

and

Key Encapsulation

Slides by Prof. Jonathan Katz.
Lightly edited by me.

Encrypting long messages

- Public-key encryption schemes “natively” defined for short messages
  - E.g., El Gamal encryption

- How can longer messages be encrypted?

Encrypting long messages

- Can always encrypt block-by-block
  - I.e., to encrypt $M = m_1, m_2, \ldots, m_\ell$, do:
    $$\mathrm{Enc}_{pk}(m_1), \ldots, \mathrm{Enc}_{pk}(m_\ell)$$
- If the underlying scheme is CPA-secure (for short messages), then this is CPA-secure (for arbitrary length messages)

- What is the size of the ciphertext?

Note

- (Public-key) encryption is NOT a block cipher
  - F_k is deterministic, one-to-one, and looks random
  - Enc_{pk} is randomized (if it is CPA-secure), thus not one-to-one, and may not look random

  $\Rightarrow$ CTR-mode/CBC-mode don't make sense for public-key encryption

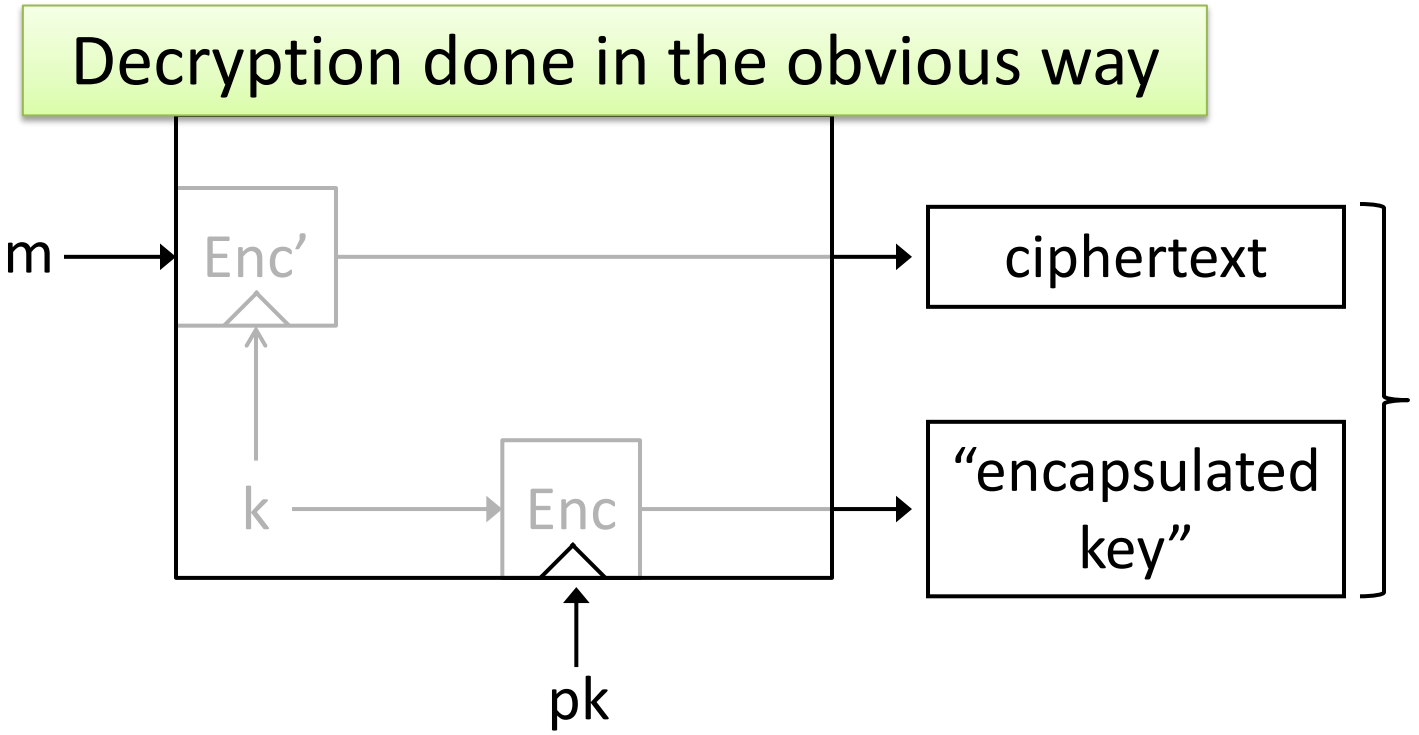
Encrypting long messages

- Encrypting block-by-block is inefficient
  - Ciphertext expansion in each block
  - Public-key encryption is “expensive”
- Can we do better?

Hybrid encryption

- Main idea
  - Use public-key encryption to establish a (shared, secret) key k
  - Use k to encrypt the message *with a symmetric-key encryption scheme*
- Benefits
  - Lower ciphertext expansion
  - Amortized efficiency of *symmetric-key* encryption

Hybrid encryption



The *functionality* of public-key encryption
at the (asymptotic) *efficiency* of private-key encryption!

Formally

- Let Π be a public-key scheme, and let Π' be a symmetric-key scheme
- Define Π_{hy} as follows:
  - Gen_{hy} = Gen (i.e., same as Π)
  - $Enc_{hy}(pk, m)$:
    - Choose $k \leftarrow \{0,1\}^n$
    - $c \leftarrow Enc_{pk}(k)$
    - $c' \leftarrow Enc'_k(m)$
    - Output c, c'
  - Decryption done in the natural way…

Security of hybrid encryption

- If Π is a CPA-secure public-key scheme, and Π' is a CPA-secure private-key scheme, then Π_{hy} is a CPA-secure public-key scheme
  - Suffices for Π' to be EAV-secure

- If Π is a CCA-secure public-key scheme, and Π' is a CCA-secure private-key scheme, then Π_{hy} is a CCA-secure public-key scheme

Application to El Gamal?

- To use hybrid encryption with El Gamal, would need to encode key k as a group element
  - Can we avoid this?

- The sender doesn't care about encrypting a *specific* key, it just needs to send a random key
  - Idea: encrypt a random group element K; define the key as k = H(K)

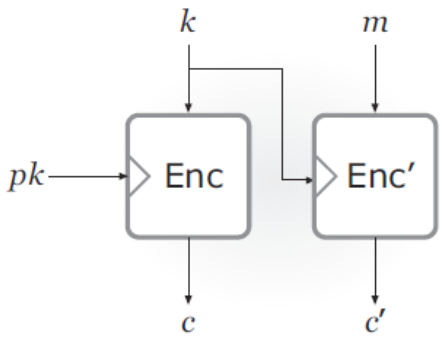
$(g^y, g^{xy} \cdot K)$

redundant?

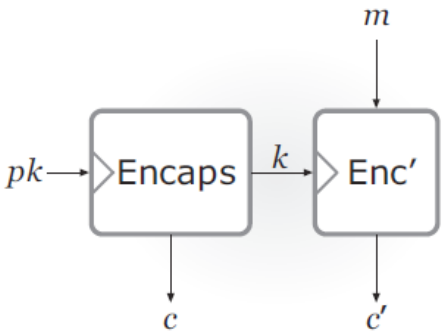
just use g^{xy}!

KEMs

- For hybrid encryption, something *weaker* than public-key encryption suffices
- Sufficient to have a “key encapsulation mechanism” (KEM) that takes a public key and outputs a ciphertext c and a key k



Hybrid encryption



KEM/DEM

KEMs

- For hybrid encryption, something *weaker* than public-key encryption suffices
- Sufficient to have a “key encapsulation mechanism” (KEM) that takes a public key and outputs a ciphertext c and a key k
  - Correctness: k can be recovered from c given sk
  - Security: k is indistinguishable from uniform given pk and c; can define CPA-/CCA-security
- Can still combine with symmetric-key encryption as before!

Security of KEM/DEM

- If Π is a CPA-secure KEM, and Π' is a CPA-secure private-key scheme, then combination is a CPA-secure public-key scheme
  - Suffices for Π' to be EAV-secure

- If Π is a CCA-secure KEM, and Π' is a CCA-secure private-key scheme, then combination is a CCA-secure public-key scheme

KEMs vs. PKE schemes

- For short messages, direct encryption using a PKE scheme (with no hybrid encryption) can sometimes be the best choice

- For anything longer, KEM/DEM or hybrid encryption will be more efficient
  - This is how things are done in practice (unless very short messages are being encrypted)

KEM based on El Gamal

- Gen(1^n)
  - Run $\mathcal{G}(1^n)$ to obtain G, q, g. Choose uniform $x \in \mathbb{Z}_q$. The public key is (G, q, g, g^x) and the private key is x
- $Ecaps_{pk}$, where pk = (G, q, g, h)
  - Choose uniform $y \in \mathbb{Z}_q$. The ciphertext is g^y, and the key is $k = H(h^y)$
- $Decaps_{sk}(c)$, where sk = x
  - Output $k = H(c^x)$

Security?

- If the DDH assumption holds, and H is modeled as a random oracle, then this KEM is CPA-secure

Complete scheme

- Combine the KEM with private-key encryption

- I.e., encryption of message m is
  g^y, $Enc'_k(m)$,
  where $k = H(h^y)$ and Enc' is a symmetric-key encryption scheme
  - If Enc' is CPA-secure and H is modeled as a random oracle, this is a CPA-secure public-key encryption scheme

Chosen-ciphertext security

- Under stronger assumptions, this approach can be proven to give CCA security
  - If Enc' is a CCA-secure symmetric-key scheme
- Can at least see why the previous attack no longer works
- Standardized as DHIES/ECIES

RSA-based KEM

- Idea: use *plain* RSA…

  …but on a random value!
- Then use that random value to derive a key

RSA-based KEM

- Encaps:
  - Choose uniform $r \in \mathbb{Z}^*_N$
  - Ciphertext is $c = [r^e \bmod N]$
  - Key is $k = H(r)$

- Decaps(c)
  - Compute $r = [c^d \bmod N]$
  - Compute the shared key $k = H(r)$

Security?

- This KEM can be proven CCA-secure under the RSA assumption, if H is modeled as a random oracle

Comparison to RSA-OAEP?

- The RSA-KEM must be used with a symmetric-key encryption scheme
- For very short messages (< 1500 bits), RSA-OAEP will have shorter ciphertexts
- For anything longer, ciphertexts will be the same length; RSA-KEM is simpler