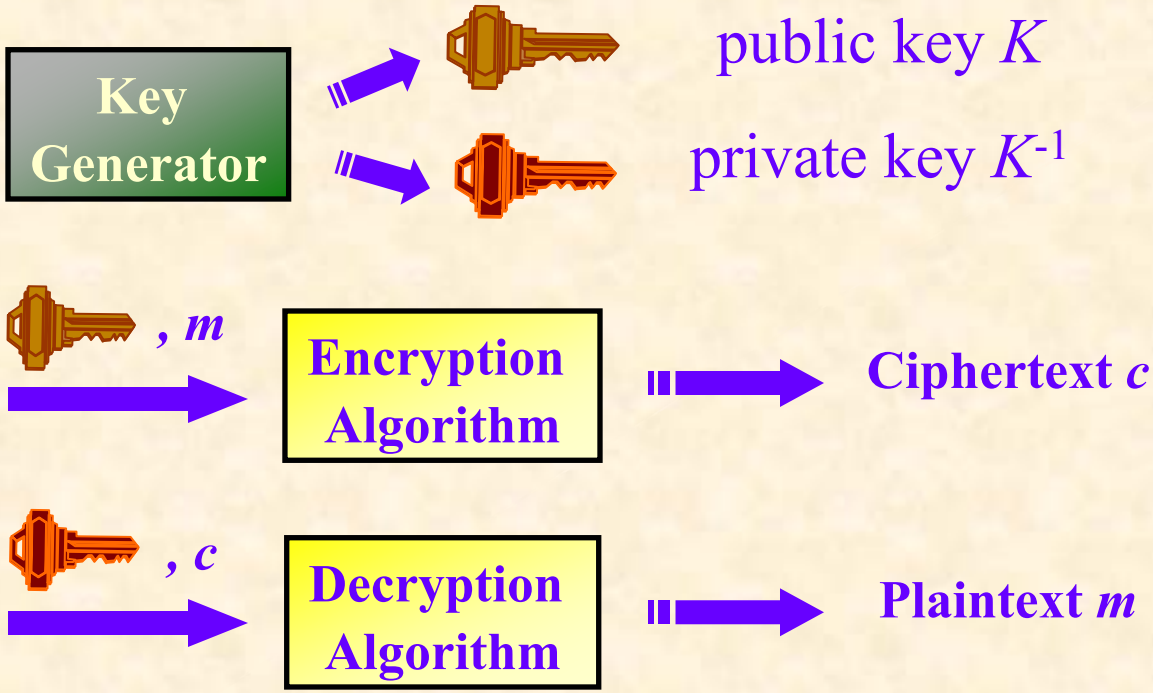
Be careful how you pad;
Your encryption scheme may be as good as the padding you use…

Tassos Dimitriou

Athens Information Technology

Definition of Public Key Encryption

- A public key encryption scheme is a triple <*G*, *E*, *D* >, where



- Encryption and decryption are inverses of each other
- If $c = E_K(m)$, then an adversary should obtain "no information" about the message m.

What does “no information” mean?

- **It is easier to think about “insecurity” than security.**
  For example:

- **Given *c*, the adversary has no idea what *m* is…**
  - Often something about message is known
- **Derivation of key from a few ciphertexts…**
  - Absense of key recovery does not make the scheme secure
- **The adversary cannot recover *m* from ciphertext…**
  - May be able to figure out partial information about m

In a secure encryption scheme, *no partial information should leak*.

Impact on Encryption

- **Furthermore, it *should not be possible to relate ciphertexts* of different messages**
  - Encryption must be *probabilistic* or *stateful*
- ***This goes against the historical notion of encryption***
  - Encryption is no longer a *fixed* mapping of plaintexts to ciphertexts.
  - A single plaintext will have many possible ciphertexts
  - Yet, it should be possible to decrypt...

Computational Security

- **Perfect security attempts to capture the notion of "how much more" the adversary learns**
  - A scheme is perfectly secure if the possession of ciphertext does not reveal any *additional* information about message
- **Perfect security is "expensive" and not practical**
  - Requires a key as long as the *total amount* of data to be encrypted.
- ***Computational security*** **is a better notion**
  - Works with adversaries of limited computing power.
  - If adversary works harder, she can learn more, but any feasible amount of effort should not reveal any noticeable information!

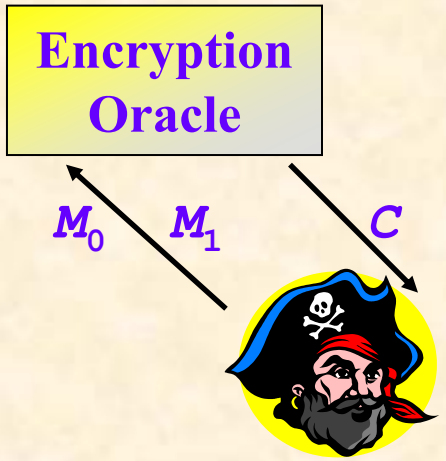
Chosen Plaintext Attack (CPA)

Motivation: Encrypted messages should look the same to computationally restricted adversaries.

"Worst case" for security: Sender wants to encrypt one of two known messages.

Model: Adversary sends **two** messages to an ***encryption*** **oracle**

- Oracle returns the ciphertext of one of them.
- Adversary is allowed to choose message pairs via a CPA attack.



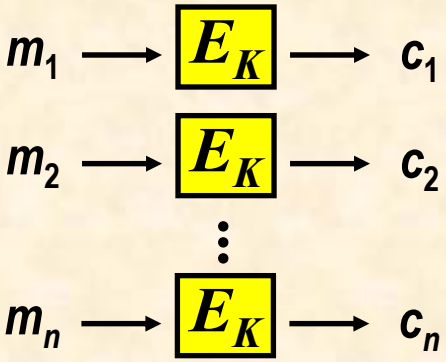
Goal: Tell which of the two "worlds" the oracle lives in.

- The scheme is ***CPA-secure*** if adversary has a hard time telling which message was encrypted.

Example of a CPA attack

- Let E: $\{0,1\}^k \times \{0,1\}^L \rightarrow \{0,1\}^L$ be a block cipher, used to encrypt messages of length L.
- *Electronic Code Book* (ECB) is a mode for encrypting messages $M = <m_1, m_2, \ldots m_n>$ that consist of many blocks.

Encryption of $m = <m_1, m_2, \ldots m_n>$



Decryption is defined in the natural way

- Is this CPA-secure? No because same blocks encrypt to same ciphertexts.
- Encryption must be probabilistic or stateful!

Cipher-Block Chaining (CBC)

- A useful mode of operation that can be shown to be CPA-secure is Cipher-Block Chaining (CBC).

Encrypt $(\langle m_1, m_2, \ldots, m_n \rangle)$

Let $IV =_R \{0,1\}^L$

for i=1 to n do

$c_i = E_K(m_i \oplus c_{i-1})$

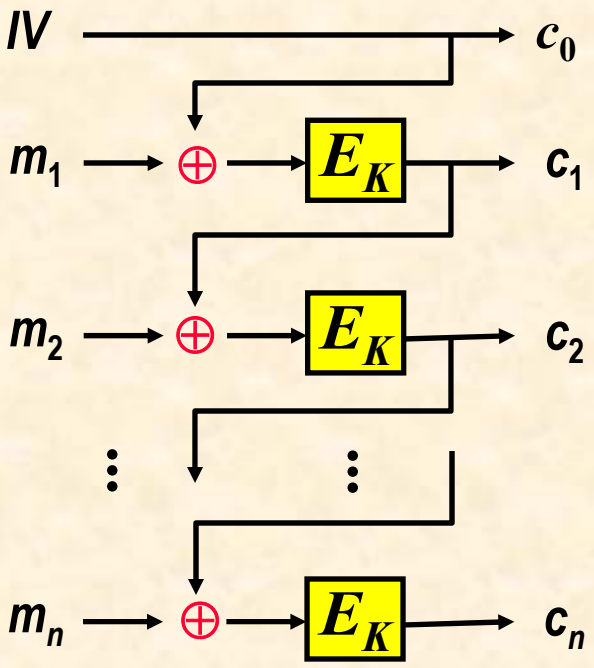
return $(\langle c_0, c_1, c_2, \ldots, c_n \rangle)$

Decrypt $(\langle c_0, c_1, c_2, \ldots, c_n \rangle)$

for i=1 to n do

$m_i = E_K^{-1}(c_i) \oplus c_{i-1}$

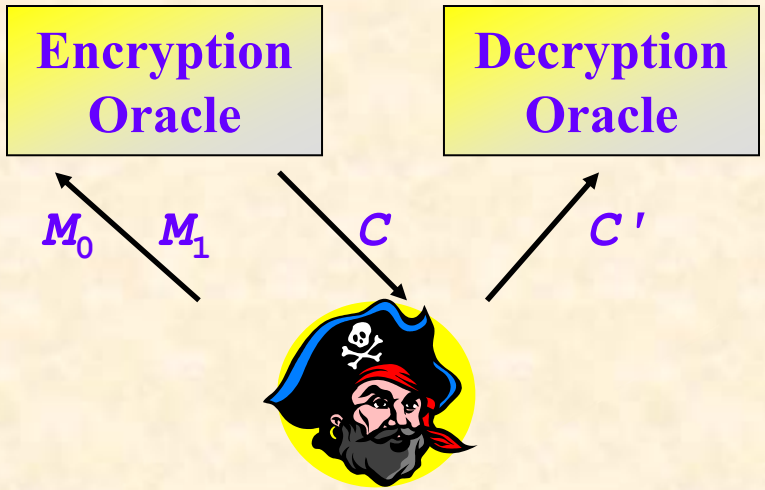
return $(\langle m_1, m_2, \ldots, m_n \rangle)$



Chosen Ciphertext Attack (CCA)

- **Sometimes we want to consider privacy when the adversary is capable a stronger type of attack, namely a *chosen ciphertext attack.***
- **Now the adversary has access to a *decryption* oracle. How can this be possible?**
  - Adversary gains temporary access to decryption equipment
  - Model attempts to capture the security of new ciphertexts in the face of previous access to decryption oracle.
  - Authenticated key exchange protocols are prone to such attacks.
- **Furthermore, this situation may arise unexpectedly through *side channels*!**

CCA Security



- **Adversary submits two messages to encryption oracle.**
  - Oracle returns the ciphertext C of one of them.
  - Adversary is *not* allowed to ask decryption oracle about C
  - She can ask, however, about *modified* versions of C
- **The scheme is *CCA-secure* if adversary has a hard time telling which message was encrypted.**
- **CCAttacks can break all standard modes of operation (CBC,…)**

Side Channel Attacks

- Having access to a decryption oracle maybe *unrealistic*.
- An adversary, however, can exploit side channels which return enough information about a ciphertext to be decrypted easily.

Side channels arise frequently in practice by giving the ability to an adversary to induce predictable changes to plaintexts through modification of the ciphertext.

- We will review some of these attacks on both *symmetric* and *asymmetric* encryption schemes and propose ways do prevent such attacks.

Attacks on Symmetric schemes
[Vaudenay 2002]

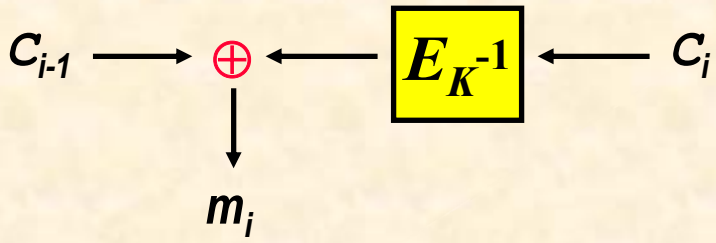
- In the description of CBC, the length of a message must be a multiple of the block length *L*. When this is not the case, *padding* must be used.
- CBCPAD is a byte oriented padding scheme which takes the form “01”, “02 02”, “03 03 03”, “04 04 04 04”, etc.

Q: What should the receiver do after decryption if he discovers that the padding is *not* valid?

  - This depends on the protocol used
  - SSL/TLS specify that the session be torn down
  - ESP just logs the error, WTLS returns an error message
- If adversary can ascertain the padding error status, it can use it as a side channel to mount a CCA attack.

Attack on CBC

- Model this behavior as a *padding oracle* that returns VALID if plaintext is correctly padded, otherwise INVALID.
- Recall decryption formula: $m_i = E_K^{-1}(c_i) \oplus c_{i-1}$



Fixing c_i and flipping any bit of c_{i-1} flips the corresponding bit of message block m_i.

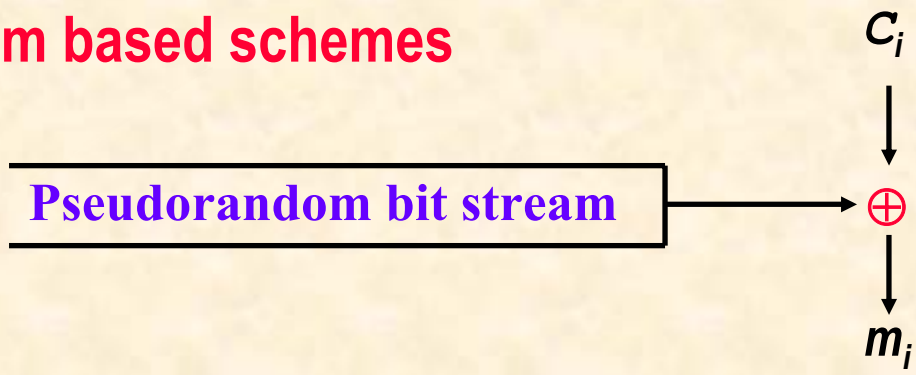
- In particular, if the ciphertext consists of the two blocks $<c_0,c_1>$, then any changes to c_0 (= IV) will be mirrored in m_1.

Attack on CBC (cont.)

- The attack works in two phases (initially m_1 is a correctly padded block of plaintext) :
- Phase 1: Randomly flip bits in IV until the Oracle says VALID
  - This happens if induced message block m_1' has a proper CBCPAD ("01" or "02 02" or "03 03 03", etc).
  - The most probable case is when the pad is 01…
- Phase 2: Once we know the final byte in m_1', we can find the final byte in m_1. This is simply the final byte of IV' $\oplus$ 01.
- Then we iterate to find the rest of the bytes in the block. Once we recover a full block, we can use the corresponding ciphertext as an IV for the next block…

Other padding methods

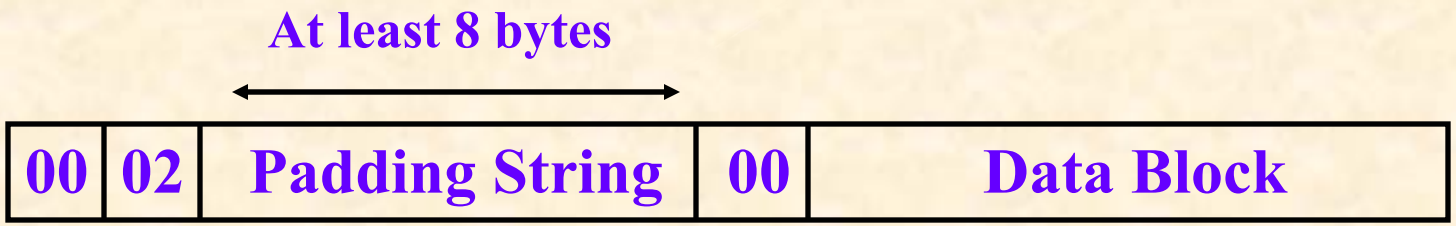
- **ESP Padding (IPSec):** If we have to pad $p>0$ bytes, we append the bytes 01, 02 … up to p.
- **XY Padding:** Uses two distinct public values X, Y
- **Obligatory 10* Padding:** Append a 1-bit to message and then zero or more 0-bits.
  - Attacks no longer apply. Virtually all messages are correctly padded.
  - Only one plaintext block is invalid under this padding: 0^n
- **Stream based schemes**



Side Channels in PK Systems

[Bleichenbacher '98, Manger '01]

- Side channel attacks are not only typical of symmetric systems but of Public Key systems as well.
- Assume the attacker has access to an oracle that returns a bit telling whether the ciphertext corresponds to data encrypted according to RSA standard PKCS #1 (v1.5)



- On the receiving end, receiver parses block from left to right to see if it is PKCS #1 *conforming*.

Side Channels in PK Systems

- **Using an oracle that tells whether a ciphertext is PKCS#1 conforming, one can break this RSA encryption scheme using about 1 million queries.**

Q: **But how does an attacker can get access to such an oracle?**

- **Plain encryption**
  - Alice sends a PKCS#1 encrypted message (e.g. a secret key) to Bob without any integrity checks.
  - The RSA encryption standard mentions that an integrity check should be used but only for signing….
- **Detailed error messages**
- **Timing attacks for applications combining encryption and signatures.**

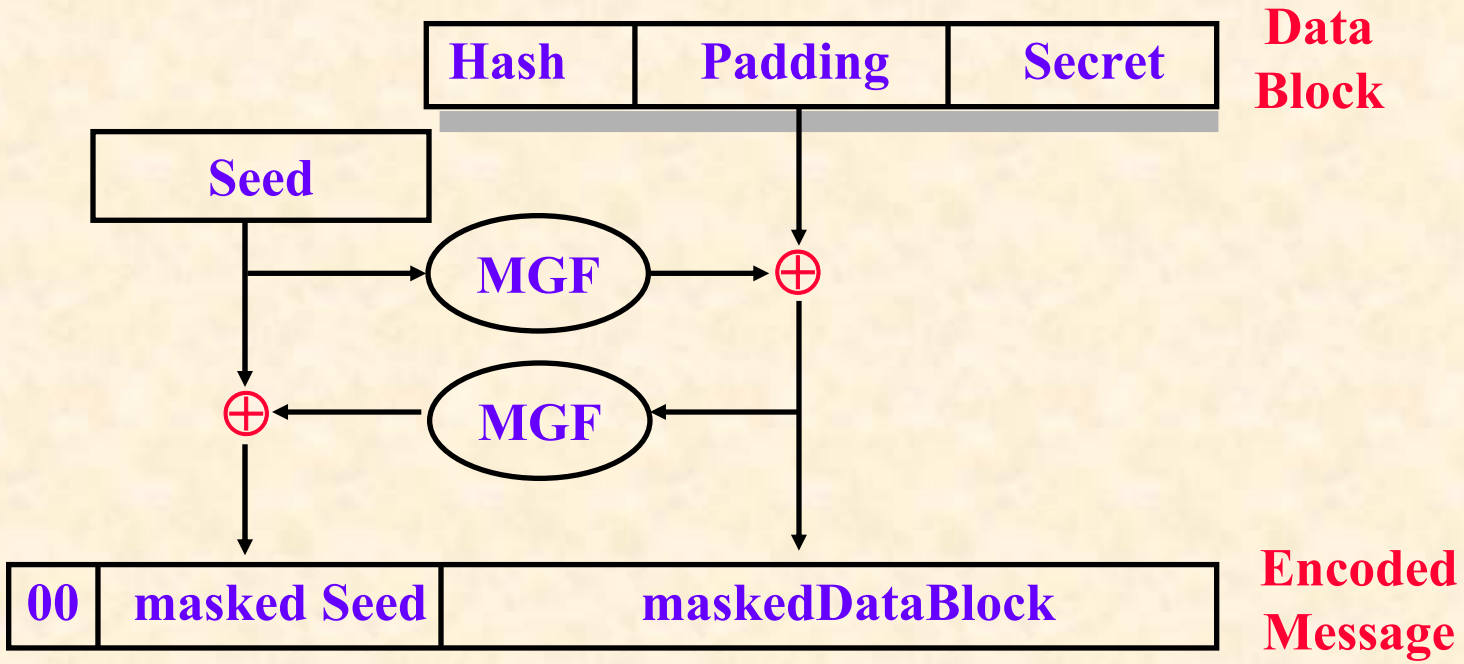
Optimal Asymmetric Encryption Padding (OAEP)

- It is important to include a strong integrity check into RSA encryption
- The phase between decryption and integrity verification is critical as any leak of information may present a security risk.
- Version 2 of PKCS #1 introduced a new algorithm RSA-OAEP that uses *Optimal Asymmetric Encryption Padding (OAEP)* to counteract the previous attack.

If a deterministic public-key encryption primitive (e.g. RSA) is hard to invert without the private key, the corresponding OAEP-based encryption scheme is *plaintext aware*.

RSA-OAEP

- Plaintext awareness is closely related to the resistance of the scheme against Chosen Ciphertext attacks.



- The encoded message is *converted to an integer* which is then encrypted using RSA.

RSA-OAEP (cont.)

- The integrity of the ciphertext is verified by comparing the hash and the padding during the decryption process.
- However, the design of RSA-OAEP makes it highly likely that implementations will leak information between the decryption and integrity check operations.

The attack starts with an assumption that the attacker can distinguish a failure in the *integer-to-octets* conversion from any subsequent failure.

- PKCS#1 recognizes this by explicitly stating that error messages during the decoding process be the same.

Likelihood of Susceptibility

- **Spelling**
  - Trivialities may differentiate error messages
- **Logs**
  - Reveal detailed error conditions
  - Are available to a much larger set of people
- **Other error conditions that may indicate that the decoding stage has been reached.**
  - Through attacker defined MGF functions.
- **Timing**
  - Even identical responses may be distinguished if they take different amount of times to occur (e.g. comparing an attacker defined hash value with that in the encoded message)

Authenticated Encryption

- **It is an old recommendation that encryption should always be accompanied by authentication.**
  - If it were impossible for the adversary to produce valid ciphertexts, all attacks would vanish…
- **A notion of security strictly stronger than CCA-security is the following mouthful: *integrity of ciphertexts with semantic security against Chosen Plaintext Attacks*. This is "authenticated encryption".**
- **Using authenticated encryption we are guaranteed that**
  - An adversary will not be able to manipulate a given ciphertext to a new valid one.
  - Nor she will be able to combine old ciphertexts…
  - The only valid ciphertexts are the one she has seen in traffic...

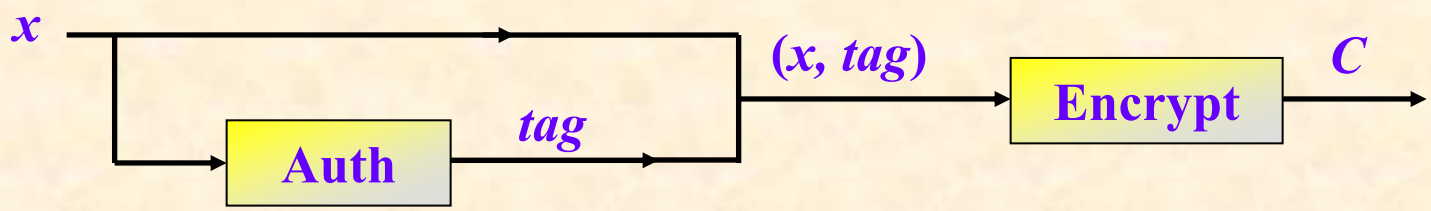
What happens in real life?

- The most widespread application of cryptography in the Internet today is for implementing a *secure channel* between two end points.
- First establish secret keys, then use these to encrypt and authenticate transmitted information.

SSL	*(Authenticate then Encrypt)* $tag = Auth\ (x),\quad C = Encrypt\ (x,\ tag),\qquad$ transmit C
SSH	*(Encrypt and Authenticate)* $C = Encrypt\ (x),\quad tag = Auth(x),\qquad transmit\ (C,\ tag)$
IPSec	*(Encrypt then Authenticate)* $C = Encrypt\ (x),\quad tag = Auth(C),\qquad transmit\ (C,\ tag)$

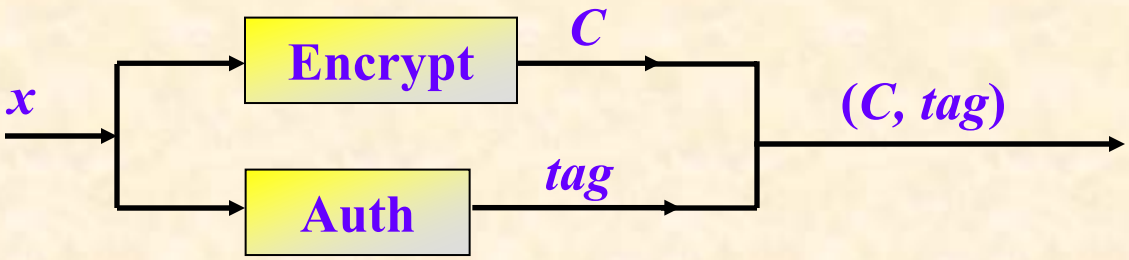
- Is there a "right way"?

SSL - Authenticate *then* Encrypt



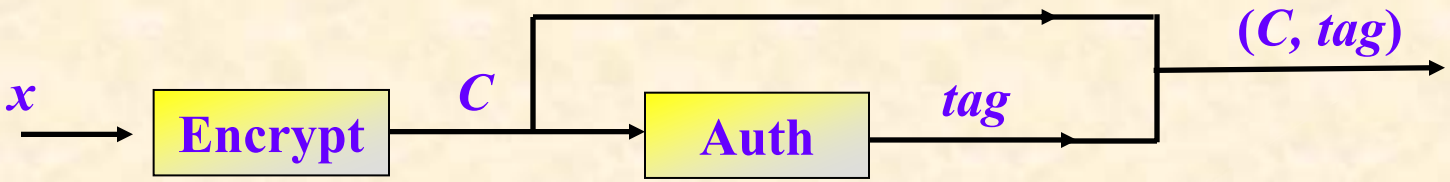
- ***AtE* method is *not* generically secure!**
  - Even with a perfect encryption function and a perfect MAC
  - Attack is not against authenticity of information but against secrecy
  - Intuition that changes to ciphertext will be discovered by underlying MAC is not true.
- ***It is* secure, however, under two very common forms of encryption: CBC mode and stream ciphers (that XOR data with a pseudo-random bit sequence).**
- ***Beware* of "slight changes"**

SSH - Encrypt *and* Authenticate



- ***E&A* method is *not* generically secure!**
  - MAC can be secure against forgeries but still leak information on the plaintext.
  - In reality, attack is not very practical…
  - Can be fixed by requiring the SSH session to rekey frequently…

IPSec - Encrypt *then* Authenticate



- ***EtA* method *is* generically secure!**
  - If one applies to each transmitted message the composed function *EtA* then the secrecy and authenticity of the resulting channel is guaranteed.

- **Any secure channel protocol designed to work with *any* combination of secure encryption and secure MAC *must use* the encrypt-then-authenticate method.**

Moral

Always check authenticity first, i.e. reject inauthentic messages without any further processing

Do not (over) trust intuition, do not take security as an obvious property of anything and mind every little change to a secure method.