

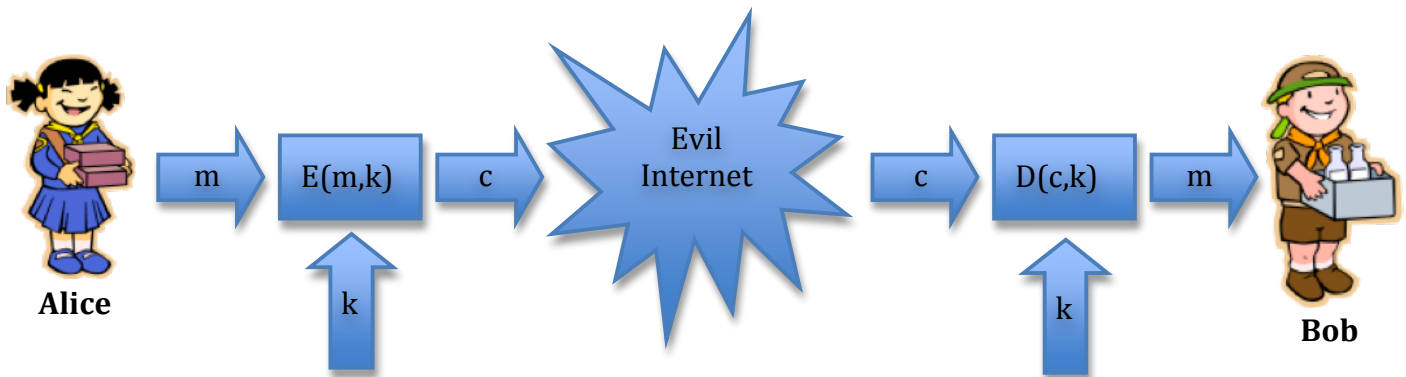
Introduction

The primary goal of Cryptography is to allow **secure communication** over an **insecure medium**.

One way to achieve secure communication would be to *create* a secure medium. This could be achieved by physically connecting the communicating parties with a physical medium, buried underground in cement, guarded by the military, etc. This solution does not scale. Unlike wires buried underground, the Internet is readily available to all, cheap, and insecure.

Symmetric Key Cryptography

In symmetric key cryptography both parties are required to share a secret (key) prior to the communication exchange. The exchange can be modeled as follows, where m =message, k =key, c =ciphertext, E =encryption function, and D =decryption function.

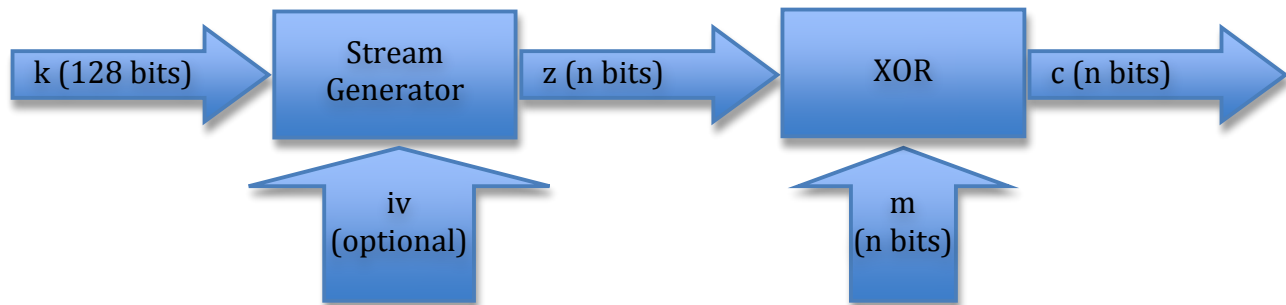


A symmetric cipher should allow an adversary to see many pairs of both message (m) and ciphertext (c) without revealing any information about the key (k). Although this may seem a surprising requirement at first, in practice it is not uncommon for an attacker to know m . One example of this was seen in WWII, in which Enigma operators often ended messages with common phrases or occasionally repeated entire messages.

Stream Ciphers

A stream cipher takes a pseudorandom number as input and uses that as a seed to produce an extremely long string of bits. Given that the input seed is large (128 bits), it should be reasonably impossible to distinguish between the bits output by the cipher and truly random bits without knowing the seed in advance.

A stream cipher can be viewed in the following diagram, in which k =key (seed), iv =initialization vector, z =stream, m =message and c =ciphertext.



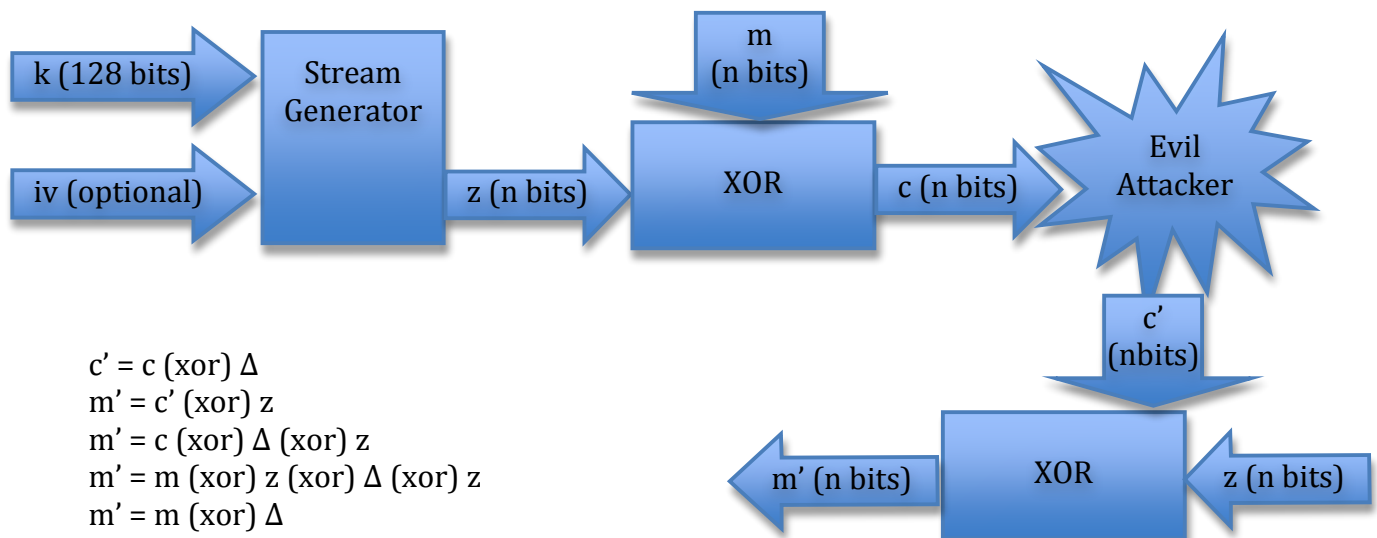
Caution: If the same stream is used for multiple transactions, then problems can result. Note the following: $(m \text{ xor } z) \text{ xor } (m' \text{ xor } z) = m' \text{ xor } m$. This means if an attacker intercepts 2 messages encrypted with the same stream, and knows what one of the messages is, he can decrypt the other.

In order to defend against this, an initialization vector can be used. This acts as a second seed input to the stream generator that prevents the same stream from being used twice. The initialization vector can be safely sent in plain text to the recipient as long as the same vector is never repeated. One possibility for ensuring the same initialization vector is never repeated is to use a sequence number.

WEP: A case study in misuse of stream ciphers

WEP has several flaws in their implementation of a stream cipher. One flaw is that sequence numbers will occasionally be reused. For instance, if a laptop hibernates and then resumes a wi-fi connection, the sequence number will be reset to zero.

Attacks on Stream Ciphers.



$$\begin{aligned}
 c' &= c \text{ (xor) } \Delta \\
 m' &= c' \text{ (xor) } z \\
 m' &= c \text{ (xor) } \Delta \text{ (xor) } z \\
 m' &= m \text{ (xor) } z \text{ (xor) } \Delta \text{ (xor) } z \\
 m' &= m \text{ (xor) } \Delta
 \end{aligned}$$

If you just encrypt content and send it over the Internet, that isn't enough. An attacker can still tamper with the data, possibly in ways that will have effects he can predict, and the recipient will be unable to detect the tampering. See the above diagram for a more complete understanding.

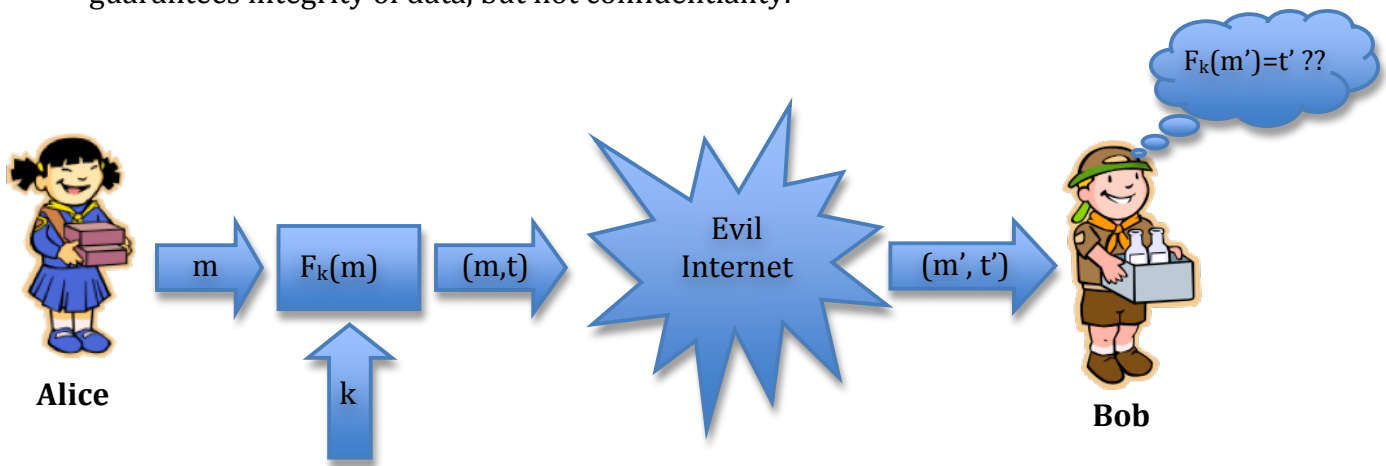
Imagine a scenario in which an attacker does not know the exact contents of a message, but has a rough idea of the data contained in the message and its relative location.

ATM: A simple example

- Bob requests \$300 from a bank account at an ATM
- The ATM contacts Alice (the bank) to inform Alice of the transaction, and ask for authorization of the amount
- Alice deducts \$300 from Bob's account and replies that the ATM should go ahead and spit out \$300
- An attacker intercepts Alice's transmission. The attacker knows the format of the data transmitted by the bank, and strategically flips a bit corresponding to the 10th bit in the amount the ATM should dispense.
- Bob (the ATM) receives a message to go ahead with the transaction and dispense \$300 + \$1024 = \$1324.

Message Authentication Codes (MAC).

In order to guarantee the integrity of data, Alice and Bob can send a Message Authentication Code (or tag) along with a message. To use a MAC both parties must share a secret function, F_k . Alice applies this function to her message, and sends the output along with the message. Bob can apply the function to the message he receives, and check that the output matches the MAC value. An attacker could change the message, but would be unable to forge the MAC since he does not know F_k . An attacker should be able to see many pairs of messages and tags, but be unable to learn anything about the function generating the tags. This approach guarantees integrity of data, but not confidentiality.



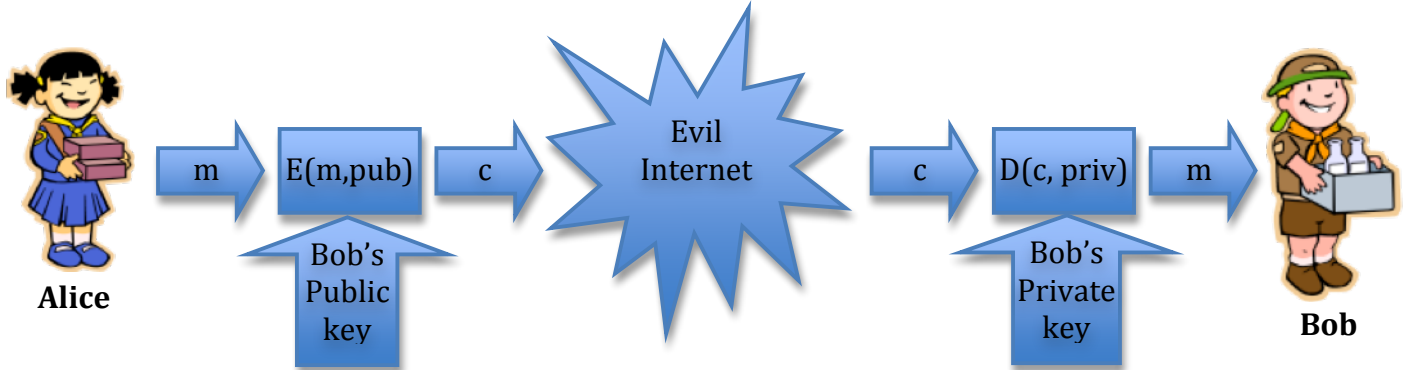
When combining integrity and confidentiality techniques, it is necessary to encrypt *first*, and *then* apply authentication techniques. Authenticating and then encrypting will normally work, but not always. Even if secrecy (and not integrity) is the only property desired, authentication techniques should still be used.

WEP: An attack on encrypted, unauthenticated data.

Consider a scenario where encryption without authentication is used to secure a wi-fi connection between a laptop and a wireless access point, as is the case with WEP. Suppose that an attacker can catch a packet being transmitted from the laptop to the router, modifies them, and feeds them back to the router. In the altered packets, the attacker could flip select bits in the packet corresponding to the destination IP address. The IP address is at a predictable location in the packet. This could change the destination of the packet to an IP address belonging to the attacker, rather than the legitimate destination. The access point will then decrypt the packet and forward the packet on to the attacker, where he can read the contents. This could be avoided if the access point authenticated the packets.

Public Key Cryptography.

In public key cryptography, separate information is used to encrypt and decrypt data. This has the advantage that two parties can exchange information without having to first share a secret.



This method of secret sharing will provide confidentiality, but not integrity. In order to provide integrity, signatures are necessary. To use a signature, Alice will compute a tag value (hash function) of her message. She will then encrypt the tag value using *her private key*. Once Bob has decrypted the message, he can then compute the same hash himself, and decrypt the tag value using *Alice's public key*. If the tag computed by Bob matches the tag value decrypted using Alice's public key, then the message is authentic.

Certificates.

Unfortunately, Alice and Bob not needing to share a secret before they can communicate with each other (as is the case with symmetric key cryptography) only

solves half of the problem. For Alice to send a message to Bob, she still needs a way to get Bob's private key that she can trust. On the Internet, this is addressed through the certificate system.

Suppose that Alice would like to purchase some goods online from Bob. Alice will decide ahead of time to trust a third party, who we'll call Verisign. At some previous date, Bob will convince Verisign of the authenticity of his identity and ask Verisign to associate a particular public key with Bob. Verisign will do this by creating a document, called a "certificate", which says "Bob's public key is xy...z" and then sign this document with Verisign's own private key. Now, when Alice wants to have a secure conversation with somebody who may or may not be Bob, she will simply ask the purported Bob for his certificate from Verisign that contains his public key. As Alice has decided to trust Verisign, and already has Verisign's public key, Alice will use Verisign's public key to decrypt the document and learn Bob's public key. Alice can now use this key to encrypt messages that she would like to send to Bob, and rest assured that only Bob's can decrypt them.

Note several properties of this exchange. First, once Verisign has signed a document containing Bob's public key, Verisign does not need to be involved in any more transactions. Bob (or any other party) can freely distribute this signed document at will. Second, Alice does not have to trust that the party who she is communicating with, and who has given her the signed document with Bob's public key, is actually Bob. She only has to trust that only Bob knows Bob's private key, Verisign has not signed a false document, and the key which Alice has for Verisign is correct.

Cryptography Running Times in Practice.

	Public Key Algos.		Symmetric Key Algos.		
	RSA	DSA	AES	MD5	SHA1
Encrypt	30K/sec	null	150MB/sec	null	null
Decrypt	15K/sec	null	150MB/sec	null	null
Sign	1.5K/sec	3K/sec	150MB/sec	50-550 MB/sec	null
Verify	30K/sec	2.5K/sec	150MB/sec	50-550 MB/sec	null
Hash	null	null	null	null	40-400 MB/sec

Timing measurements taken on a standard desktop computer. Decrypting tends to take longer because it requires raising a value to an unguessable value (ie. private key) that therefore must be large.

Secure Channels.

One interesting and useful challenge is to establish a secure channel for communication using the security primitives developed. This is harder than it may seem. Consider an approach in which Alice and Bob send messages, and each

message is both encrypted and authenticated. This would be vulnerable to the following attacks:

Replay: A attacker blindly parrots transmissions from one party to another multiple times. Although the party cannot know the contents, the recipient of the replayed messages will be unable to differentiate between messages sent by the attacker and those sent by the true sender. Several potential defenses for attacks of this type include unique identifiers sent with each message. One option is for Alice to timestamp messages, or include sequence numbers if Alice and Bob are unable to establish a synchronized clock. A second option is for Bob to send Alice a random number called a “Nonce” before each transmission, which Alice then includes in her message.

Reordering: An attacker may intercept and reorder messages, and the recipient will be unable to detect any change.

Repeat: An attacker can play Alice’s messages back to Alice as if they have come from Bob.

Traffic Analysis: An attacker cannot tell the content of messages, but he can tell who is sending, who is receiving, how often transmissions occur and how long transmissions are. This can reveal more information than one would guess. Consider an attacker monitoring a secure connection to a website. Based on the timing and direction of traffic, the attacker can likely guess which packet represents a URL request. The length of the request will leak information about which page is likely being requested. Next, the attack can monitor the traffic as the page loads. If a page is known to have a certain amount of text, and certain images of certain sizes, that may be enough to identify the page.