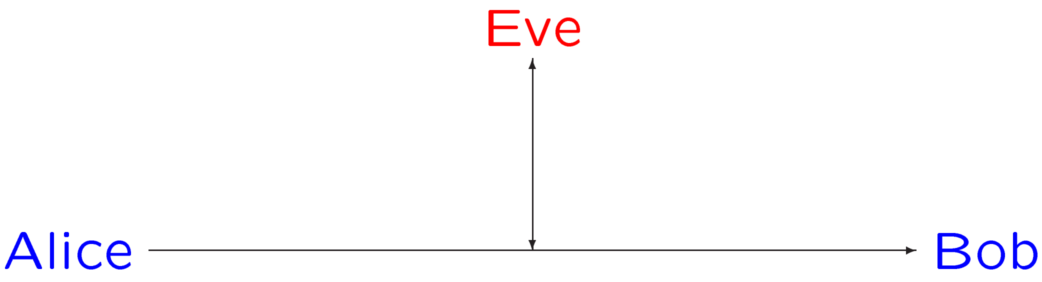
Practice-Oriented Provable-Security:

How to protect yourself against un-anticipated attacks

Mihir Bellare

UC San Diego

Cryptography is about how to communicate in the presence of an adversary.

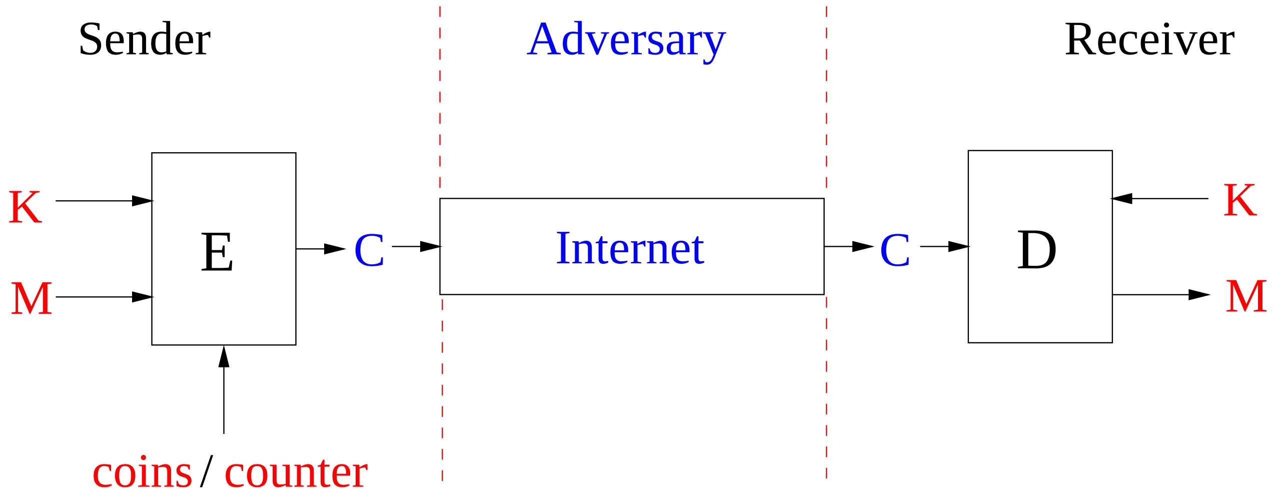


Example problems: Ensure privacy and authenticity of transmitted data.

The job of a cryptographer is to design protocols. View these as means to create envelopes that specify how to

- encapsulate data before transmission
- decapsulate received information to recover the data.

Envelopes



$\mathcal{E}$ = Encapsulation or encryption procedure that takes the key K and plaintext M to produce a ciphertext C

$\mathcal{D}$ = Decapsulation or decryption procedure that takes the key K and ciphertext C to recover the plaintext M

The history of envelopes

- **Stone age of cryptography:** Shift or Substitution ciphers
- **Mechanical period:** The Enigma machine; the one-time pad
- **Modern times:** computers and mathematics; a clear notion of a key.

Example: In the shift cipher (used by Julius Caesar) each letter of the alphabet is shifted by some amount, eg. 4 below:

$$\begin{array}{cccccc} A & B & C & D & \cdots & Z \\ \downarrow & \downarrow & \downarrow & \downarrow & \cdots & \downarrow \\ E & F & G & H & \ldots & D \end{array}$$

Atomic primitives

We have some advantages over Julius Caesar:

- RSA – Rivest, Shamir Adleman function
- Block ciphers – eg. DES

RSA is a function f

$$x \mapsto f(x) = x^e \bmod N$$

that is one-way: easy to compute, hard to invert as long as the problem of factoring large numbers is hard.

DES specifies for any 56-bit key K a map DES_K taking 64 bit inputs to 64 bit outputs, whose behavior looks "random" if you don't know K.

Security of atomic primitives

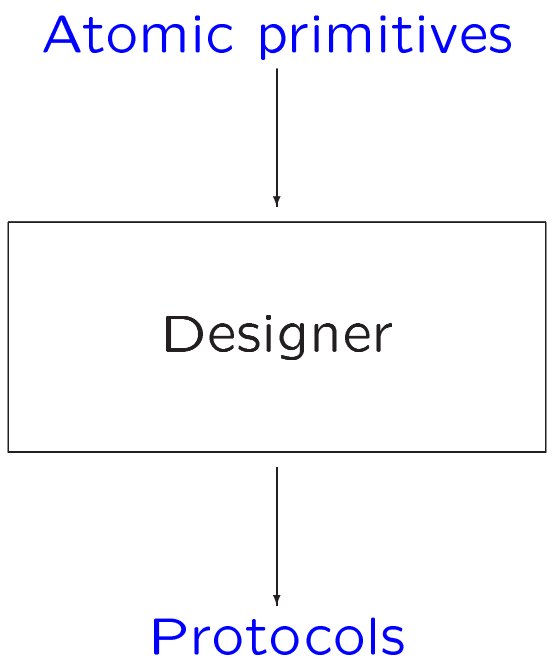
Atomic primitives like DES, RSA are taken to be secure.

How do we know this?

Largely faith. They have resisted attack until now so we believe in them. We have little choice, since we have to get off the ground somehow.

So (if we have secure atomic primitives) what's there to do?

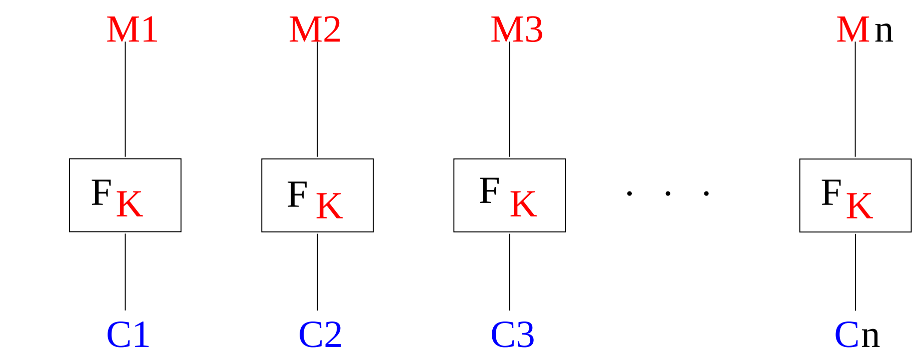
Atomic primitives by themselves don't solve any cryptographic problem of interest. They must be appropriately used.



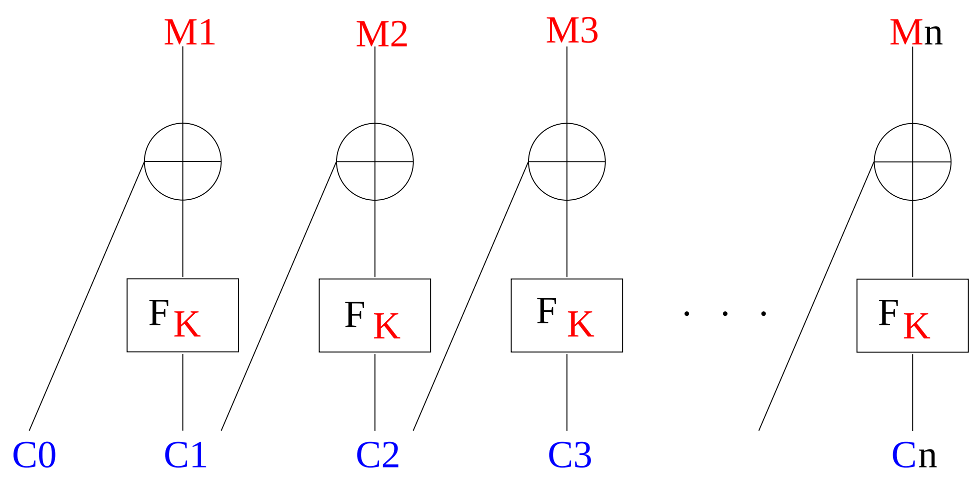
For example, how do you encrypt a text tile? DES only processes 64 bits.

Solutions may start springing to mind. For example, let $F =$ DES and try

ECB Encryption:



CBC Encryption:



But one must be careful. Which of the above are really good?

Typical existing practice: Iterative design

Problem

$\downarrow$

Protocol 1.1

$\downarrow$

Bug!

$\downarrow$

Protocol 1.2

$\downarrow$

Bug!

$\vdots$

$\vdots$

Protocol 1.n

$\downarrow$

Ship it!

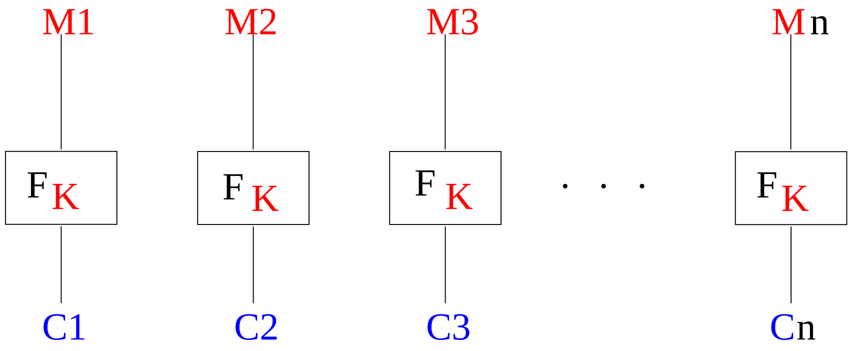
$\downarrow$

Bug!

The problem

Bugs (security holes) appear because atomic primitives are mis-used. The protocol is insecure even though the underlying atomic primitive is secure.

For example, it turns out ECB encryption is insecure. There is an attack: if two message blocks are the same it is detected.



There is no point having a strong lock on the door if you leave the window open. We want to design protocols that preserve the strength of primitives, not loose it.

Classical view: Cryptographic protocol design is a battle between designer and cryptanalyst.

But we cannot enumerate or anticipate all the attack strategies that an attacker might try. Testing will fail as a means to ensure security because we cannot test against every possible attack.

Thesis: An inability to find attacks is not strong evidence of the security of a protocol.

Classical cryptography is aware of this, but what else can you do other than test for attacks?

Question: Is there some way to guarantee security even against un-anticipated attacks?

Provable security

Theory can help!

Since the 80s, there has developed a theory of security for cryptographic protocols called provable security, whereby one can get a proof that a protocol is really secure against all attacks.

Looks promising ... but ...

This theory was originally not too practical.

Over the last few years we have been developing a practice oriented provable security. This yields practical, proven secure protocols.

Provable security

An idea of Goldwasser and Micali [GM82]

- Make formal definitions of security
- Provide a reduction: A proof that the only way to break the protocol is to break the underlying atomic primitive.

Bug? Implies the basic primitive can be broken!

Example

Thm: [BDJR97] If F is a "secure" block cipher, then CBC encryption is a provably "secure" encryption scheme.

Translation: If a hacker can break CBC encryption, then, using the hacker as a subroutine, I can quickly and easily break F.

But the world's best cryptographers do not know how to easily break F, so we don't expect the hacker to be able to break CBC encryption.

Provable security

Bad news for security consultants.

To "classical" cryptographers, the concept can be difficult to accept. They view cryptography as a battle between the designer and the cryptanalyst. When you have provable security, there is no need for cryptanalysis of the protocol.

Provable security might at first glance seem impossible. We will however see examples of how it is done.

Our work and its impact

- Analysis of CBC encryption [BDJR97] and MAC [BKR94]
- Design of OAEP encryption scheme [BR94]– used in RSA PKCS#1 v2.0; proposed for IEEE P1363a
- Design of PSS signature scheme [BR96]– proposed for IEEE P1363a
- Design of HMAC [BCK96]– Internet Standard for the IETF IPSEC, implemented in many places including SSL, B-SAFE, ...

Corporations ask —and even pay!— for proofs of security

Our treatment of block ciphers has influenced the design criteria of upcoming ciphers like AES.

Pause

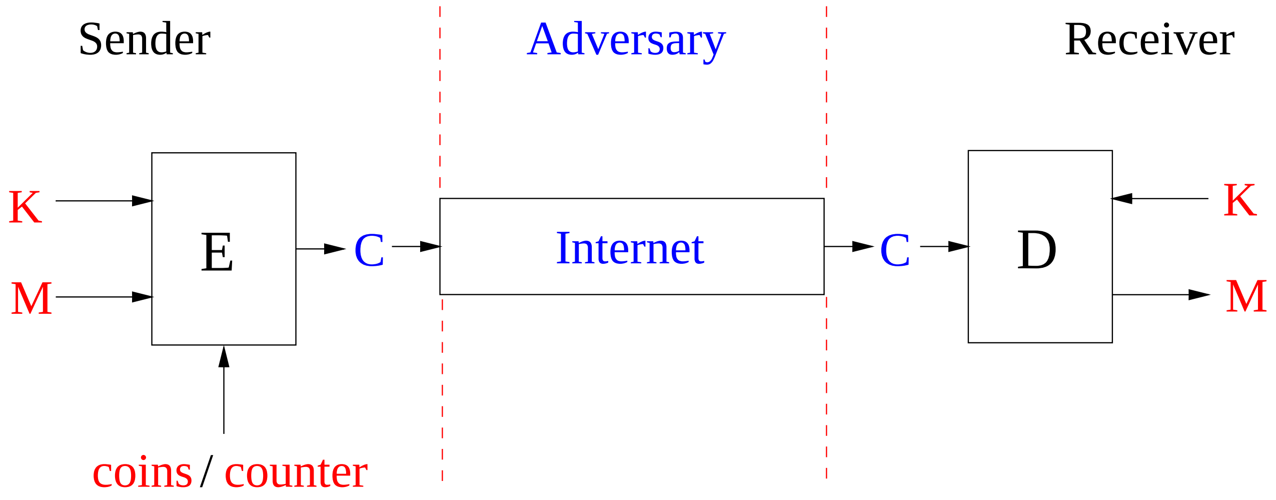
Coming up next:

I will illustrate the approach for the problem of symmetric encryption, following

> M. Bellare, A. Desai, E. Jokipii and P. Rogaway. A Concrete Security Treatment of Symmetric Encryption: Analysis of the DES Modes of Operation. Extended abstract in *Proceedings of FOCS 97*. Full version available at my web page.

We will go through definitions and reductions.

An encryption scheme



$\mathcal{E}$ = Encapsulation or encryption procedure that takes the key K and plaintext M to produce a ciphertext C

$\mathcal{D}$ = Decapsulation or decryption procedure that takes the key K and ciphertext C to recover the plaintext M

We have not said anything about security; this is just about the algorithmic interface.

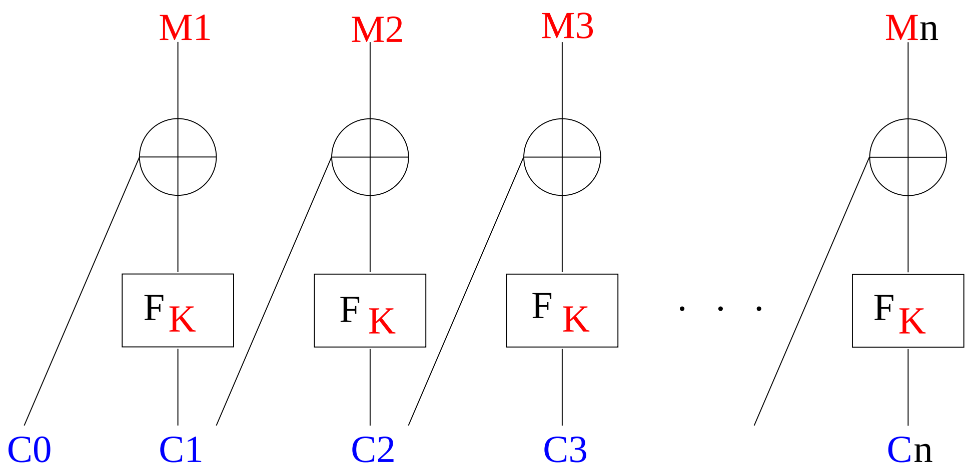
CBC Encryption

Suppose we want to encrypt a n-block text

$$M = M_1 \dots M_n \, .$$

Let F be a block cipher (eg. DES).

Pick a random IV C_0 and:



$\mathcal{E}_K(M)$ is $C_0 C_1 \dots C_n$.

Question: Is this a secure encryption scheme?

When is an encryption scheme secure?

Is a scheme secure if: Given the ciphertext $C = \mathcal{E}_K(M)$ it is hard to find the key K?

Not enough! For example, the encryption scheme that transmits the message in the clear has this property but is certainly **not** secure.

When is an encryption scheme secure?

Is a scheme secure if: Given a ciphertext $C = \mathcal{E}_K(M)$ it is hard to find the plaintext M?

Certainly required, but not enough either. For example, remember that in ECB mode, if two message blocks are the same it is detected.

We want instead that even partial information about the plaintext not be leaked [GM,BDJR].

What is a secure encryption scheme?

Idea of a definition: Consider a game:

- Adversary chooses two different messages M_0, M_1.
- Give the adversary $E_K(M_b)$ where b is a random bit
- Ask the adversary to guess which message was encrypted

If adversary can win game not much more often than half the time, say the scheme is secure.

Adversary is allowed a chosen-plaintext attack; can request encryptions of messages of its choice.

Turns out this captures a very strong notion of security.

The notion

Possibly the central component of the provable security approach is that we have formal definitions of security. They give us a way to pin down what we are looking for.

Takes some getting used to.

For example: we agreed that from the ciphertext you should not be able to recover the plaintext. Where is this reflected in our strange, abstract game above?

It is, because if it were easy to get the plaintext from a ciphertext then we could easily win the game.

A consequence

Encryption must be probabilistic or stateful.

Ie. if you encrypt many times the same message M, you must get out different ciphertexts each time.

Why? Formally, otherwise you could win the game. Intuitively, otherwise, if the sender is only encrypting "yes" and "no", you can tell which.

CBC encryption is probabilistic due to the IV C_0.

Quantitative security

We want to assess how well CBC encryption meets this notion.

Let Adv(Eve) be the “advantage” over 1/2 with which E can win the game.

Assume μ bits are encrypted. Let l be the block length of the cipher. (Eg. 64 for DES). We will measure Adv(Eve) as a function of μ, l and the “strength” ϵ of the block cipher.

This ϵ describes the probability of breaking the cipher as a function of the number of known plaintext / ciphertext pairs and time used by the adversary.

Security of CBC: Birthday attack

Let l be the block length of the cipher. (Eg. 64 for DES).

Once we encrypt $\mu = l \cdot 2^{l/2}$ bits of data, CBC encryption becomes insecure, *no matter how strong the cipher*.

This is due to a birthday attack which wins our game against CBC encryption when $\mu = l \cdot 2^{l/2}$.

The attack does not find the key!

More generally we are saying there is some Eve so that

$$\mathrm{Adv}(\mathrm{Eve}) \geq 0.3 \cdot \frac{\mu^2}{l^2} \cdot \frac{1}{2^l}\,.$$

Security of CBC: Main result

Theorem: [BDJR] Let Eve be any adversary who asks the oracle for the encryptions of messages totaling μ bits. Then

$$\mathrm{Adv}(\mathrm{Eve}) \leq 2\epsilon + \frac{\mu^2}{l^2} \cdot \frac{1}{2^l},$$

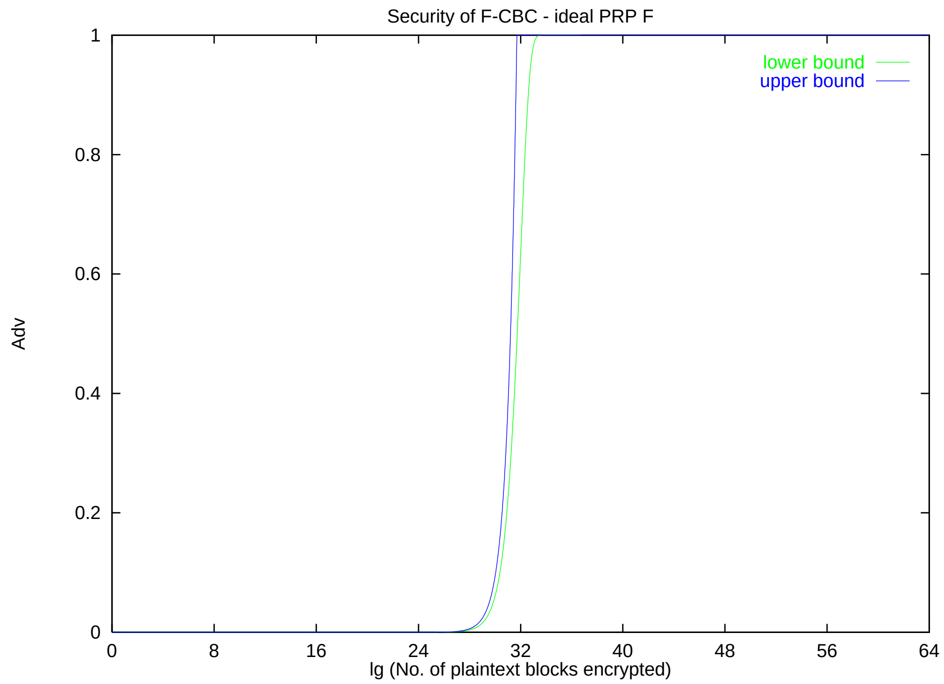
where ϵ is the probability that Eve can "break" the block cipher given $q = \mu/l$ plaintext-ciphertext pairs and about the same running time as Eve.

We know: ϵ depends on the key size of the cipher. For example for DES, with key size 56, ϵ is very small when q is less than 2^{32}.

Translation: Given that you can't break DES with 2^{32} plaintext-ciphertext pairs, CBC encrypting up to 2^{32} blocks with a single DES key is fine.

Plot of security of CBC

Assume block cipher is “ideal”.



Lower bound is the attack.

Upper bound is proven security.

Security of a block cipher

What exactly do we mean by "breaking" a block cipher?

Not key recovery.

We ask that the block cipher be a finite pseudorandom function / permutation family [GGM,BKR], meaning breaking it is detecting some "deviation" from random behavior, given some plaintext / ciphertext pairs, and without knowing the key.

This is a general and useful viewpoint [BKR].

Security of CBC

Summary: Chance of breaking the scheme is

$$2\epsilon + \Theta\left(\frac{(\#\text{ plaintext blocks encrypted})^2}{2^{\text{block length of cipher}}}\right),$$

where ϵ measures strength of underlying block cipher.

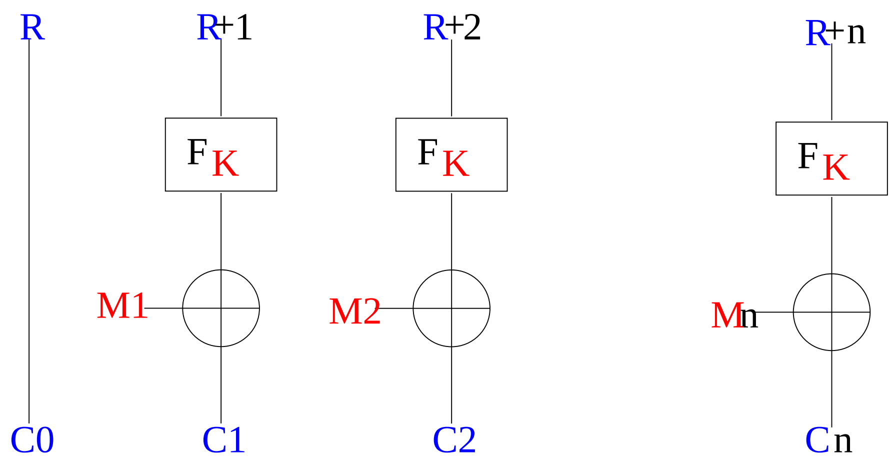
Things to notice:

- Block size can be more relevant than key size
- In particular, triple DES CBC also becomes insecure after 2^{32} blocks are encrypted.

Conclusion: In designing future block ciphers, increase the block size, not just the key length. AES is doing this.

CTR (aka XOR) Mode

Sender maintains l-bit counter R, incremented by n after an n-block message is enciphered.



- Don't need to compute F_K^{-1} to decrypt
- Receiver does not need to store the counter.
- Permits parallelism and off-line encryption.

Security of CTR

Theorem: [BDJR] Let Eve be any adversary who asks the oracle for the encryptions of messages totaling $\mu < l2^l$ bits. Then

$$\mathrm{Adv}(\mathrm{Eve}) \leq \epsilon$$

where ϵ is the probability that one can "break" the block cipher given $q = \mu/l$ plaintext-ciphertext pairs and about the same running time as Eve.

Translation: Even if you encrypt 2^{64} blocks under a single key then CTR encryption is perfectly secure in a strong sense, as long as you use a pseudorandom function F_K.

"Break" the block cipher means detect deviation in behavior from a random function (non-invertible cipher).

Proof paradigm

These theorems need proofs.

Lemma 1: If F_K is replaced by a truly random function in the CBC scheme, then

$$\mathrm{Adv}(\mathrm{Eve}) \leq \frac{\mu^2}{l^2} \cdot \frac{1}{2^l},$$

no matter how high is the running time of Eve.

Lemma 2: If Eve can break CBC under F with probability more than

$$2\epsilon + \frac{\mu^2}{l^2} \cdot \frac{1}{2^l}$$

then, given the above, it must be that F has easily detectable non-random behavior.

Intuition: Why is CBC good?

If F_K is replaced by a truly random function in the CBC scheme, and if there are no collisions in values of the random function, then Eve has zero advantage in winning our game.

So the advantage of Eve is bounded by the probability of collisions.

What POPS buys

Attacks tell us whether a protocol is insecure, practice oriented provable security can tell us it is secure.

Practice oriented provable security is (pretty much) the **ONLY** way to get security guarantees for high level protocols. No amount of attack based analysis will give you this!

You can get confidence in a protocol via reductions, not via attacks.

POPS complements attack based analysis. For example in case of CBC encryption:

- Insecure if more than 2^{32} message blocks are encrypted (birthday attacks)
- Secure if less than this are encrypted

Components of POPS

DEFINITIONS ; REDUCTIONS ; PROOFS

Having definitions makes it easy to find attacks and bugs as well as analyses.

Reductions give the security guarantees by showing: "only way to break protocol is to break underlying primitive."

Proofs show that reductions are correct.

Summary

High quality crypto is important for security. Yet real world protocol design tends to be ad hoc and often low quality.

Theory can help! Via the notion of provable security.

But the application is not immediate. We are pursuing

PRACTICE ORIENTED provable security

which has already had some success.