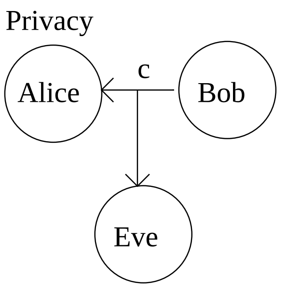
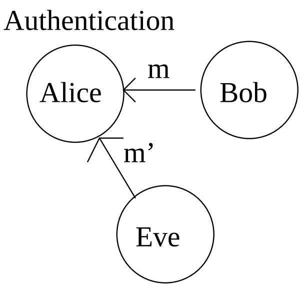
6.875J/18.425J Cryptography and Cryptanalysis	November 26, 2001
Lecture 21	
Guest Lecturer: Moses Liskov	*Scribed by: Jennifer Mulligan*

1 Recap

Throughout the term, we have been considering two major cryptographic problems: privacy and authentication. In the privacy problem, Bob wishes to send a message to Alice in such a way that she can read it but the evesdropper (Eve) cannot. In the authentication problem, Bob wishes to be able to send a message to Alice in an authenticated way, so Alice knows that he sent the message. Here, an adversary should not be able to make a message that looks as if it was sent from Bob.

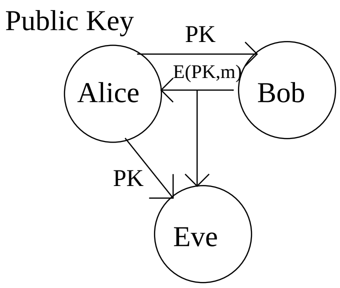




There are two models we have considered: the public-key model and the symmetric-key model.

1.1 Public-key model

In the public key model, one party (Alice for encryption, Bob for authentication) has a public key PK and a secret key SK. The public key is published and its secrecy need not be protected. The other party need not have any secret information at all.

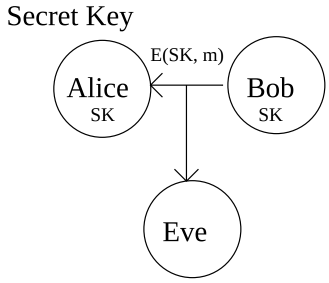


In the public-key model, *public-key cryptosystems* are the solutions to the privacy problem and *digital signature schemes* are the solutions to the authentication problem. We have seen a number of examples of actual public-key encryption schemes: RSA, Goldwasser-

Micali, ElGamal, Cramer-Shoup. We haven't seen as many examples of digital signature schemes, but we have seen a couple: Merkle-Lamport, for instance, and GMR.

1.2 Symmetric-key model

In the symmetric-key model, Alice and Bob share a secret key that they use to encrypt or authenticated messages. Eve does not know this secret key and if the system is secure, she cannot figure out the message either.



Here, symmetric-key encryption schemes are the solution to the privacy problem, and MACs are the solution to the authentication problem. We have seen only one example of a symmetric-key encryption scheme - the one-time pad, and we have seen only one example of a MAC - using a PRF. Both of these are inadequate solutions. The one-time pad provides absolute privacy but the fact that it is one-time severely limits its use.

Similarly, PRFs provide only a limited form of MACs. First of all, they cannot be used to authenticate messages of arbitrary length, and second, they produce outputs that are too large.

1.3 Block ciphers

In this lecture, we give better constructions for symmetric-key cryptosystems and for MACs. What we have to start with is a building block: pseudorandom permutations, also known as block ciphers.

Recall the definition: *A family of permutations $P.(\cdot)$ is pseudorandom if both P and P^{-1} are efficiently computable and the output is indistinguishable from a random permutation, given only oracle access to P_s. More precisely, let $\Pi(1^k)$ be the uniform distribution of all permutations over $\{0,1\}^k$. Then $P.(\cdot)$ is a pseudorandom permutation if $\forall$ PPTF $\{A_k\}$, there exists a negligable function $\nu(k)$, such that*

$$Pr[s \leftarrow \{0,1\}^k; \mathcal{O}_0 = P_s; \mathcal{O}_1 \leftarrow \Pi(1^k);$$
$$b \leftarrow \{0,1\}; b' \leftarrow {A_k}^{\mathcal{O}_b}(1^k) : b = b'] \leq 1/2 + \nu(k)$$

2 Symmetric-key encryption

2.1 Definition

First of all, we need a definition of secure symmetric-key encryption. We can't apply the definition we used when we proved the one-time pad secure, since that was defined with an unbounded adversary in mind.

What level of security should we consider? We have studied two notions of security for public-key cryptosystems: the standard notion, and adaptive chosen ciphertext attack, where the adversary has access to a decryption oracle. In the symmetric-key model, things are a little different. Since there is no "public key" to give out, the basic notion doesn't carry much security. However, it seems reasonable to give the adversary access to oracles. The attack models we could consider are:

- Adaptive chosen plaintext attack: the adversary has access to an encryption oracle.
- Adaptive chosen ciphertext attack: the adversary has access to both an encryption oracle and a decryption oracle.

For the time being, we'll focus on security against adaptive chosen plaintext attack.

Secure symmetric-key encryption against adaptive chosen plaintext attack

$\forall PPTF\{A_k\}\exists\nu \in negl\forall k,$
$Pr[s \leftarrow \{0,1\}^k; (m_0, m_1, \alpha) \leftarrow {A_k}^{E_s(\cdot)}(k); b \leftarrow \{0,1\}; c \leftarrow E_s(m_b); b' \leftarrow {A_k}^{E_s(\cdot)}(\alpha, c) :$
$b = b'] \leq 1/2 + \nu(k)$

This gives a definition which is essentially a version of one-pass GM security in the symmetric-key model. However, one slight adjustment should be made. In the real world, we are not interested in encrypting only messages of one particular size. We wish, instead, to encrypt messages of whatever size we choose. However, it is impossible to be completely secure when messages can be this arbitrary – for one thing, longer messages simply must take more space than shorter messages, so an adversary could tell messages apart based on their lengths. It is reasonable to say, though, that this is the only victory we will permit. So, we define security as follows:

$\forall PPTF\{A_k\}\exists\nu \in negl\forall k,$
$Pr[s \leftarrow \{0,1\}^k; (m_0, m_1, \alpha) \leftarrow {A_k}^{E_s(\cdot)}(k); b \leftarrow \{0,1\}; c \leftarrow E_s(m_b); b' \leftarrow {A_k}^{E_s(\cdot)}(\alpha, c) :$
$b = b' \wedge |m_0| = |m_1|] \leq 1/2 + \nu(k)$

Now, let's go about trying to construct a secure symmetric-key cryptosystem from a PRP.

2.2 First attempt

Suppose we have a PRP, P_s, where $|s| = |x| = |P_s(x)| = k$. Let m be a message to encrypt. Break m up into k-bit blocks: $m = m_1 \circ m_2 \circ \ldots \circ m_l$.

Given a secret key s, we encrypt by computing $c_i = P_s(m_i)$ for all $i \in [1, l]$, where $c = c_1 \circ c_2 \circ \ldots \circ c_l$ is the encryption of m. This is called the "electronic code book" mode of operation, ECB for short.

Unfortunately this scheme is not secure because it is deterministic. Therefore one can easily determine which of the two messages was encrypted by running the encryption algorithm on both of them.

2.3 Second attempt

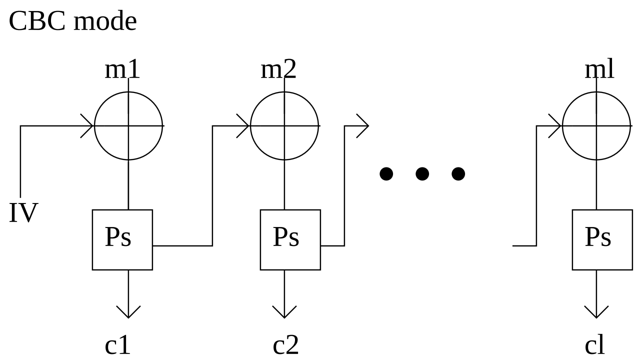
It's clear that we need to do something random in order to achieve security with block ciphers. Thus, we try using a random value to offset things. We use a random "initialization vector" IV. Now, instead of computing $c_i = P_s(m_i)$, we let $c_i = P_s(m_i \oplus IV)$, and let $c = (c_1 \circ \ldots \circ c_l, IV)$.

This is actually secure when we only encrypt one-block messages, but for longer messages, the scheme can be broken using a simple attack. Initially, the adversary gives the two messages $m_0 = 0^k 0^k$ and $m_1 = 0^k 1^k$. If the ciphertext is two repeated strings, then we know it was m_0, and otherwise the message was m_1. The weakness is that the same IV is used for all blocks of the encryption.

Now, we could fix this by using a different random value for each block of the message, but this is grossly inefficient – the size of the ciphertext would be twice the size of the message. We can do better.

2.4 CBC mode

Here is a third attempt. This time, let $c_1 = P_s(m_1 \oplus IV)$ and let $c_i = P_s(m_1 \oplus c_{i-1})$ for $i > 1$. Let $c = (c_1 \circ \ldots \circ c_l, IV)$ as before. This is called the "cipher block chaining" mode, CBC for short.



This time, we do get a symmetric-key cryptosystem secure against adaptive chosen plaintext attack. We will prove that if this is not secure, then P_s is not a PRP.

Proof: Suppose we have an oracle $\mathcal{O}$, and suppose A_k is an adversary that defeats the security of the cryptosystem. We can use $\mathcal{O}$ as if it were the function P_s and thus simulate the function $CBC{-}{-}\mathcal{O}$. If $\mathcal{O}$ is in fact P_s then $CBC{-}{-}\mathcal{O}$ will be just like a real encryption. If, on the other hand, $\mathcal{O}$ is a random permutation, $CBC - -\mathcal{O}$ will behave differently.

1. $(\alpha, m_0, m_1) \leftarrow {A_k}^{CBC--\mathcal{O}}(k)$
2. $b \leftarrow \{0, 1\}$
3. $c = CBC - -\mathcal{O}(m_b)$

4. $b' \leftarrow {A_k}^{CBC--\mathcal{O}}(\alpha, c)$
5. If $b' = b$, return "PRP" else return "random."

Our probability of sucess is then:
$Pr[S_{PRP}(A'(k))] = 1/2 * Pr[S_{PRP}(A'(k))|\mathcal{O} = P_s] + 1/2 * Pr[S_{PRP}(A'(k))|\mathcal{O} = \pi]$

Lemma 1: We have an advantage when $\mathcal{O} = P_s$
We know this because when the oracle is P_s, we succeed with the same probability as A_k which is $1/2 + 1/poly$.

Lemma 2: When the oracle is random, we have a probability of 1/2 - $\nu(k)$ of sucess.

The idea here is that if we never give $\mathcal{O}$ the same input twice then we might as well decide which message the output is an encryption of after the adversary finishes computing. If this is the case, then the probability of success must be 1/2.

However, if A_k makes some number $p(k)$ of queries to $CBC--\mathcal{O}$, then there can be at most $l \cdot p(k)$ queries to $\mathcal{O}$ invoked. Furthermore, every query to $\mathcal{O}$ is random unless there has been a collision. Every time the adversary asks for the encryption of a message m, we generate an independent, random IV, and query $\mathcal{O}$ on input $m \oplus IV$, which if $\mathcal{O}$ is a random permutation, will give us a random value (since $m \oplus IV$ has a uniform distribution, regardless of m.) Thus, there are at most $l^2 \cdot p(k)^2$ random queries to $\mathcal{O}$, and so there is at most a probability $l^2 \cdot p(k)^2/2^k$ of any collisions occurring. This is a negligible probability. Since the worst case for our proof is if the adversary is correct whenever possible if $\mathcal{O}$ is random, we note that with probability at least $1/2 - (l^2 \cdot p(k)^2)/2^k$, A_k is incorrect when $\mathcal{O}$ is random.

Therefore we know that $Pr[S_{PRP}(A'(k))] \geq 1/2 + 1/poly$ and so P_s cannot be a PRP. Therefore since P_s is a PRP, this encyrption scheme is secure.

Thus, we have achieved a scheme which is secure against adaptive chosen plaintext attack.

2.5 Security against CCA

Unfortunately, if we allowed a chosen ciphertext attack on this encryption system, it would no longer be secure. We can easily see this that given an encryption (c, IV) we can ask for $D(c, IV \oplus 1)$ that gives us m'. Then we can get back the original message m from $m' \oplus 1$.

3 MACs

3.1 Definition

A function family $\{M_s(\circ)\}$ is a MAC if for all PPTF $\{A_n\}$, there exists a negligible function $\nu(k)$ such that

$$\Pr[s \leftarrow \{0,1\}^k; (m, x), Q \leftarrow A_k^{M_s} : x = M_s(m) \text{ and } (m, x) \notin Q] \leq \nu(k)$$

3.2 Block-cipher based MAC

We can use a PRF as a MAC but there are several difficulties to make it work. The first issue is that a PRF may have an output of any length and it may even be longer than the message itself. An ideal MAC would only add a constant number (proportional to the security parameter) of bits to the message so that the message doesn't grow much longer. Another difficulty is that a particular PRF only accepts messages of one length, therefore we could not create a MAC for any message we wish.

So one idea here is to use a PRP of a fixed length (say k bits to k bits) and then collapse a message of arbitrary size to a MAC of k bits, by using the PRP in CBC mode and just giving the last block c_l of the ciphertext as output. We don't even need to give IV since we don't have to be random anymore. We might as well always set $IV = 0^k$. Is this secure? Suppose for the moment we only authenticate messages of length kl, for a constant l.

We prove that if this MAC is not secure, then P_s is not a PRP. Suppose we are given an oracle $\mathcal{O}$ and wish to determine whether $\mathcal{O}$ is a PRP or a random permutation. Our adversary A' works as follows using an adversary A_k that breaks the MAC:

1. $(m, x, Q) \leftarrow {A_k}^{CBCMAC--\mathcal{O}}$
2. If $x = CBCMAC - -\mathcal{O}(m)$ and $(m, x) \notin Q$, return "PRP" else return "random."

Our probability of sucess is then $Pr[S_{PRP}(A'(k))] = 1/2 * Pr[S_{PRP}(A'(k))|\mathcal{O} = P_s] + 1/2 * Pr[S_{PRP}(A'(k))|\mathcal{O} = \pi]$.

Claim 1: Our probability of sucess when $\mathcal{O} = P_s$ is $1/2 + 1/poly$.
Our probability of success here is equal to $p + 1/2(1-p)$ where p is the probability that A_k is successful. Since A_k is successful with nonnegligible probability, our probability here is at least $1/2 + 1/poly$.

Claim 2: Our probability of sucess when $\mathcal{O} = \Pi$ is $1/2 - negl$.

The overview of the proof of this claim is as follows. If the input for the last block, $c_{l-1} \oplus m_l$, was never queried before, the result would be totally random, so we'd only be right with negligible probability. The adversary would not be able to distinguish that it had been given a random oracle instead of the proper one. However, the adversary cannot necessarily force this input to be anything he wants, since he doesn't have access to $\mathcal{O}$ directly.

Here, we hand-wave a bit. Basically, if the adversary successfully forges a pair (m, x) where the adversary never queried $CBCMAC - -\mathcal{O}$ on m, there must be a prefix $m_1 \circ m_2 \circ \ldots \circ m_j$ which was the shortest prefix such that it is not a prefix of any query the adversary did make. (One such prefix must exist or the adversary did not forge for a new message.)

Now, starting with c_j, each output is a truly random value since the adversary never made $\mathcal{O}$ be queried on these values before[1]. Thus, the probability that the block $m_l \oplus c_{l-1}$ was one which had been given to $\mathcal{O}$ before would be negligible. Since it is bad for us if the adversary ever manages to forge a message, let us assume that if $m_l \oplus c_{l-1}$ had been queried before, the adversary succeeds with probability 1, and otherwise succeeds with negligible probability. This ends up with a total success probability which is negligible, so

[1]This is where the handwaving occurs

the probability that our reduction succeeds, by a similar argument as in the symmetric-key encryption case, is at least $1/2 + 1/poly$.

3.3 Messages of different lengths

However, this scheme does not work adequately if we expect the MAC to work for messages of different lengths. Here is an attack which takes advantage of this:

We query m_1 to get $P_s(m_1)$, then we query the oracle for $P_s(m_1)$ and get $P_s(P_s(m_1)))$. We then answer that $(m_1 \circ 0^k, P_s(P_s(m_1)))$ is a valid message/MAC pair because 0^k will not change the value of $P_s(m_1)$ before it enters P_s for the second time.

So how can we manage to make our MAC work for messages of arbitrary length? We give three secure ways:

1. Let the key be a pair (s, s'), and output $P_{s'}(CBCMAC--P_s)(m)$.
2. Output $CBCMAC--P_s(m \circ |m|)$.
3. Let $s' \leftarrow P_s(|m|)$. Then, output $CBCMAC--P_{s'}(m)$.

To give a quick, intuitive argument for the third solution, this is indistinguishable from using different random keys for each length. If we can break the third scheme then either we can differentiate between the pseudorandom keys s' and random ones (breaking P), or we can break the CBC-MAC construction for some particular length, which is also impossible if we assume P_s is a PRP.