CPSC 367: Cryptography and Security

Michael J. Fischer

Lecture 11
February 21, 2019

ElGamal Cryptosystem

Message Integrity and Authenticity

Symmetric Digital Signatures

ElGamal Cryptosystem

A variant of DH key exchange

A variant protocol has Bob going first followed by Alice.

Alice	**Bob**
	Choose random $y \in \mathbf{Z}_{\phi(p)}$.
	$b = g^y \bmod p$.
	Send b to Alice.
Choose random $x \in \mathbf{Z}_{\phi(p)}$.	
$a = g^x \bmod p$.	
Send a to Bob.	
$k_a = b^x \bmod p$.	$k_b = a^y \bmod p$.

ElGamal Variant of Diffie-Hellman Key Exchange.

Comparison with first DH protocol

The difference here is that Bob completes his action at the beginning and no longer has to communicate with Alice.

Alice, at a later time, can complete her half of the protocol and send a to Bob, at which point Alice and Bob share a key.

Turning D-H into a public key cryptosystem

This is just the scenario we want for public key cryptography. Bob generates a public key (p, g, b) and a private key (p, g, y).

Alice (or anyone who obtains Bob's public key) can complete the protocol by sending a to Bob.

This is the idea behind the ElGamal public key cryptosystem.

ElGamal cryptosystem

Assume Alice knows Bob's public key (p, g, b). To encrypt a message m:

- She first completes her part of the key exchange protocol to obtain numbers a and k.
- She then computes $c = mk \bmod p$ and sends the pair (a, c) to Bob.
- When Bob gets this message, he first uses a to complete his part of the protocol and obtain k.
- He then computes $m = k^{-1}c \bmod p$.

Combining key exchange with underlying cryptosystem

The ElGamal cryptosystem uses the simple encryption function $E_k(m) = mk \bmod p$ to actually encode the message.

Any symmetric cryptosystem would work equally well.

An advantage of using a standard system such as AES is that long messages can be sent following only a single key exchange.

A hybrid ElGamal cryptosystem

A hybrid ElGamal public key cryptosystem.

- As before, Bob generates a public key (p, g, b) and a private key (p, g, y).
- To encrypt a message m to Bob, Alice first obtains Bob's public key and chooses a random $x \in \mathbf{Z}_{\phi(p)}$.
- She next computes $a = g^x \bmod p$ and $k = b^x \bmod p$.
- She then computes $E_{(p,g,b)}(m) = (a, \hat{E}_k(m))$ and sends it to Bob. Here, $\hat{E}$ is the encryption function of the underlying symmetric cryptosystem.
- Bob receives a pair (a, c).
- To decrypt, Bob computes $k = a^y \bmod p$ and then computes $m = \hat{D}_k(c)$.

Randomized encryption

We remark that a new element has been snuck in here. The ElGamal cryptosystem and its variants require Alice to generate a random number which is then used in the course of encryption.

Thus, the resulting encryption function is a *random function* rather than an ordinary function.

A random function is one that can return different values each time it is called, even for the same arguments.

Formally, we view a random function as returning a probability distribution on the output space.

Remarks about randomized encryption

With $E_{(p,g,b)}(m)$ each message m has many different possible encryptions. This has some consequences.

An advantage: Eve can no longer use the public encryption function to check a possible decryption.

Even if she knows m, she cannot verify m is the correct decryption of (a, c) simply by computing $E_{(p,g,b)}(m)$, which she could do for a deterministic cryptosystem such as RSA.

Two disadvantages:

- Alice must have a source of randomness.
- The ciphertext is longer than the corresponding plaintext.

Message Integrity and Authenticity

Protecting messages

Encryption protects message confidentiality.

We also wish to protect message integrity and authenticity.

- *Integrity* means that the message has not been altered.
- *Authenticity* means that the message is genuine.

The two are closely linked. The result of a modification attack by an active adversary could be a message that fails either integrity or authenticity checks (or both).

In addition, it should not be possible for an adversary to come up with a forged message that satisfies both integrity and authenticity.

Protecting integrity and authenticity

Authenticity is protected using symmetric or asymmetric digital signatures.

A *digital signature* (or MAC) is a string s that binds an individual or other entity A with a message m.

The recipient of the message *verifies* that s is a *valid signature* of A for message m.

It should hard for an adversary to create a valid signature s' for a message m' without knowledge of A's secret information.

This also protects integrity, since a modified message m' will not likely verify with signature s (or else (m', s) would be a successful forgery).

Symmetric Digital Signatures

Message authentication codes (MACs)

A *Message Authentication Code* or *MAC* is a digital signature associated with a *symmetric (one-key) signature scheme*.

A MAC is generated by a function $C_k(m)$ that can be computed by anyone knowing the secret key k.

It should be hard for an attacker, without knowing k, to find any pair (m, ξ) such that $\xi = C_k(m)$.

This should remain true even if the attacker knows a set of valid MAC pairs $\{(m_1, \xi_1), \ldots, (m_t, \xi_t)\}$ so long as m itself is not the message in one of the known pairs.

Creating an authenticated message

Alice has a secret key k.

- Alice protects a message m (encrypted or not) by attaching a MAC $\xi = C_k(m)$ to the message m.
- The pair (m, ξ) is an *authenticated message*.
- To produce a MAC requires possession of the secret key k.

Verifying an authenticated message

Bob receives an authenticated message (m', ξ'). We assume Bob also knows k.

- Bob verifies the message's integrity and authenticity by verifying that $\xi' = C_k(m')$.
- If his check succeeds, he *accepts* m' as a valid message from Alice.
- To verify a MAC requires possession of the secret key k.

Assuming Alice and Bob are the only parties who share k, then Bob knows that m' came from Alice.

Cheating

Mallory *successfully cheats* if Bob accepts a message m' as valid that Alice never sent.

Assuming a secure MAC scheme, Mallory can not cheat with non-negligible success probability, even knowing a set of valid message-MAC pairs previously sent by Alice.

If he could, he would be able to construct valid forged authenticated messages, violating the assumed properties of a MAC.

Computing a MAC

A block cipher such as AES can be used to compute a MAC by making use of CBC or CFB ciphertext chaining modes.

In these modes, the last ciphertext block c_t depends on all t message blocks $m_1, \ldots, m_t$, so we define

$$C_k(m) = c_t.$$

Note that the MAC is only a single block long. This is in general much shorter than the message.

A MAC acts like a checksum for preserving data integrity, but it has the advantage that an adversary cannot compute a valid MAC for an altered message.

Protecting both privacy and authenticity

If Alice wants both privacy and authenticity, she can encrypt m and use the MAC to protect the ciphertext from alteration.

- Alice sends $c = E_k(m)$ and $\xi = C_k(c)$.
- Bob, after receiving c' and ξ', only decrypts c' after first verifying that $\xi' = C_k(c')$.
- If it verifies, then Bob assumes c' was produced by Alice, so he also assume that $m' = D_k(c')$ is Alice's message m.

Another possible use of a MAC

Another possibility is for Alice to send $c = E_k(m)$ and $\xi = C_k(m)$. Here, the MAC is computed from m, not c.

Bob, upon receiving c' and ξ', first decrypts c' to get m' and then checks that $\xi' = C_k(m')$, i.e., Bob checks $\xi' = C_k(D_k(c'))$

Does this work just as well?

In practice, this might also work, but its security *does not follow* from the assumed security property of the MAC.

The problem

The MAC property says Mallory cannot produce a pair (m', ξ') for an m' that Alice never sent.

It does *not* follow that he cannot produce a pair (c', ξ') that Bob will accept as valid, even though c' is not the encryption of one of Alice's messages.

If Mallory succeeds in convincing Bob to accept (c', ξ'), then Bob will decrypt c' to get $m' = D_k(c')$ and incorrectly accept m' as coming from Alice.

Example of a flawed use of a MAC

Here's how Mallory might find (c', ξ') such that $\xi' = C_k(D_k(c'))$.

Suppose the MAC function C_k is derived from underlying block encryption function E_k using the CBC or CFB chaining modes as described earlier, and Alice also encrypts messages using E_k with the same chaining rule.

Then the MAC is just the last ciphertext block c'_t, and Bob will always accept (c', c'_t) as valid.