Lecture 4:
Basic Public Key Cryptography

Thomas Johansson

Chapter 11: Basic Public Key Cryptography

- In public key cryptography we replace the use of identical keys with two keys, one public and one private.
- The public key can be published in a directory

Candidate One-way Functions

- We require a mathematical operation which is easy to do one way, i.e. encryption, but which is hard to do the other way, i.e. decryption, without some special secret information, namely the private key.
- Such a mathematical function is called a trapdoor one-way function, since it is effectively a one-way function unless one knows the key to the trapdoor.
- *factoring integers, computing discrete logarithms*

Problems related to factoring

You are given N but not its factors p and q.

- **FACTORING:** Find p and q.
- **RSA:** Given e such that

$$\gcd(e, (p-1)(q-1)) = 1$$

and c, find m such that

$$m^e = c \bmod N.$$

- **QUADRES:** Given a, determine whether a is a square modulo N.
- **SQRROOT:** Given a, such that

$$a = x^2 \bmod N,$$

find x.

Problems related to discrete logarithms

We are given a finite abelian group $(G, \cdot)$ and $g \in G$.

- **DLP:** Discrete logarithm problem. Given $h \in G$ such that $h = g^x$, find x.
- **DHP:** Diffie-Hellman problem. Given a, b, $a = g^x$ and $b = g^y$, find c such that
$$c = g^{xy}.$$
- **DDP:** Decisional Diffie-Hellman problem. Given a, b, c, $a = g^x$ and $b = g^y$, and $c = g^z$, determine if
$$z = xy.$$

Relations between problems

- It is important to know how they are all related
- This is done by giving complexity theoretic reductions from one problem to another.
- allows us to say that "Problem A is no harder than Problem B."
- assume an oracle (or efficient algorithm) to solve Problem B. We then use this oracle to give an efficient algorithm for Problem A.
- these reductions should be efficient, in that they run in *polynomial time*

Relations between problems

- show equivalence between two problems A and B, by showing an efficient reduction from A to B and an efficient reduction from B to A.
- If the two reductions are both polynomial-time reductions then we say that the two problems are *polynomial-time equivalent*.

Relations between problems

Lemma

In an arbitrary finite abelian group G the DHP is no harder than the DLP.

Relations between problems

Lemma

In an arbitrary finite abelian group G the DDH is no harder than the DHP.

Relations between problems

Important result:

Lemma

The FACTORING and SQRROOT problems are polynomial-time equivalent.

Relations between problems

Important result:

Lemma

The RSA problem is no harder than the FACTORING problem.

There is some evidence, although slight, that the RSA problem may actually be easier than FACTORING for some problem instances. It is a major open question as to how much easier it is.

RSA encryption

- Alice picks two large secret prime numbers p and q and computes $N = p \cdot q$.
- Alice also chooses an encryption exponent e which satisfies $\gcd(e, (p-1)(q-1)) = 1$.
- Alice's public key is the pair (N, e),
- Alice computes a decryption exponent d, satisfying

$$e \cdot d = 1 \bmod (p-1)(q-1).$$

RSA encryption

- Encryption $c = m^e \bmod N$.
- Decryption $\hat{m} = c^d \bmod N$.
- Works because

$$\hat{m} = c^d = m^{ed} = m^{1+k(p-1)(q-1)} = m.$$

RSA security

- In this chapter we shall consider security to be defined as being unable to recover the whole plaintext given the ciphertext. We shall argue in a later chapter that this is far too weak a definition of security for many applications.
- In addition in a later chapter we shall show that RSA, as we have described it, is not secure against a chosen ciphertext attack.

Lemma

If the RSA problem is hard then the RSA system is secure under a chosen plaintext attack, in the sense that an attacker is unable to recover the whole plaintext given the ciphertext.

RSA: Knowledge of $\phi(N)$ and Factoring

Lemma

If one knows the RSA decryption exponent d corresponding to the public key (N, e) then one can efficiently factor N.

RSA: Knowledge of the Private Exponent and Factoring

Lemma

Given an RSA modulus N and the value of $\phi(N)$ one can efficiently factor N.

RSA: Use of a Shared Modulus

- a number of users share the same public modulus N but use different public/private exponents, (e_i, d_i).
- Suppose Alice sends the same message m to two of the users with public keys (N, e_1) and (N, e_2),
- CAN BE BROKEN...

ElGamal Encryption

- The private key is chosen to be an integer x,
- The public key is given by $h = g^x (\mathrm{mod} p)$.

Messages are assumed to be non-zero elements of the field $\mathbb{F}_p^*$. To encrypt a message $m \in \mathbb{F}_p^*$ we

- generate a random ephemeral key k,
- set $c_1 = g^k$,
- set $c_2 = m \cdot h^k$,
- output the ciphertext as $c = (c_1, c_2)$.

Decryption

To decrypt a ciphertext $c = (c_1, c_2)$ we compute

$$\hat{m} = \frac{c_2}{c_1^x}.$$

This works because

$$\hat{m} = \frac{c_2}{c_1^x} = \frac{m \cdot h^k}{g^{xk}} = \frac{m \cdot g^{xk}}{g^{xk}} = m.$$

Example - ElGamal

Choose

$$q = 101, p = 809 and g = 3.$$

Note that q divides $p - 1$ and that g has order divisible by q in the multiplicative group of integers modulo p. The actual order of g is 808 since

$$3^{808} = 1 (\mathrm{mod} p),$$

and no smaller power of g is equal to one.

Public private key, choose $x = 68, h = g^x = 65$.

we wish to encrypt the message $m = 100$

Example - ElGamal

- We generate a random ephemeral key $k = 89$.
- Set $c_1 = g^k = 345$.
- Set $c_2 = m \cdot h^k = 517$.
- Output the ciphertext as $c = (345, 517)$.
- The recipient decrypt by $\hat{m} = \frac{c_2}{c_1^x} = \frac{517}{345^{68}} = 100$.

Security of textbook ElGamal

Lemma

Assuming the Diffie Hellman problem (DHP) is hard then ElGamal is secure under a chosen plaintext attack, where security means it is hard for the adversary, given the ciphertext, to recover the whole of the plaintext.

Proof. Suppose we have an oracle O to break ElGamal. This oracle $O(h, (c_1, c_2))$ takes as input a public key h and a ciphertext (c_1, c_2) and returns the underlying plaintext. We then show how to use this oracle to solve the DHP.

Rabin Encryption

- based on the difficulty of extracting square roots modulo $N = p \cdot q$.
- First choose prime numbers of the form $p = q = 3 \pmod 4$.
- The private key is then the pair (p, q).
- Public key: we generate a random integer $B \in \{0, 1, \ldots, N-1\}$ and then the public key is (N, B).

Rabin Encryption

- To encrypt a message m, we compute

$$c = m(m + B) \pmod{N}.$$

- Decryption is far more complicated, essentially we compute

$$m = \sqrt{\frac{B^2}{4} + c} - \frac{B}{2} \pmod{N}.$$