

**Announcements:**

1. HW #4: § 1, 2, 7, 9, 10 is due on Nov 26th
2. The next Pari-GP assignment will be announced next week
3. Read Chapter 20 [S] to get a handle on signature schemes esp. DSS and its variants.

**Topics for today:**

1. Diffie Hellman contd.
2. Massey Omura Cryptosystem
3. El Gamal Public Key System
4. Digital Signature Scheme

**1. Diffie Hellman contd.**

Our construction of Diffie-Hellman last week relied on fields with a prime number of elements. This can also be generalized to any finite field (with the # of elements being a power of a prime). From a cryptanalytic point of view, to crack this system we need to solve the so-called Diffie Hellman Problem efficiently. In what follows we can replace  $\mathbf{F}_p^*$  by any cyclic group  $G$ .

**Diffie Hellman Problem (DH).**  $\langle g \rangle = \mathbf{F}_p^*$ . Given  $g^x, g^y$  to compute  $g^{xy}$ .

Observe that this is not the discrete logarithm problem (DL). It is clear that if we solve DL we can solve DH (using  $g^x$  first computing  $x$  and then  $(g^y)^x$ ). This is denoted as  $\text{DH} \leq_P \text{DL}$  to mean DH is reducible in *Polynomial* time to DL (we'll drop the  $P$  subscript). But on the other hand whether  $\text{DL} \leq \text{DH}$  is an open problem and this stands in the way of DL being polynomial equivalent to DH. A similar question connects the RSA problem and integer factoring, one direction is clear and the other direction is an open problem.

DH is not used for encryption and is used just as a scheme to exchange private session keys. DH was the first ever *published* public key cryptosystem. Government Communications Headquarters (GCHQ) UK's security and intelligence organization claims [1] that they developed public key cryptography earlier but didn't disclose their invention.

The supposed hardness of DL can be used to build encryption and signatures systems as we'll see today.

## 2. Massey Omura Cryptosystem

All computation happens in  $\mathbf{F}_p^*$  and prime  $p$  is public.

**Initialization** in the Massey Omura (MO) system.

- $A$  secretly picks  $e_A$  and computes  $d_A$  such that  $d_A e_A \equiv 1 \pmod{p-1}$ .
- Similarly  $B$  picks  $e_B$  and computes  $d_B$  such that  $d_B e_B \equiv 1 \pmod{p-1}$ .

**Protocol** for Alice sending message  $m$  to Bob.

- Alice sends  $m^{e_A}$  to Bob.
- Bob sends  $(m^{e_A})^{e_B}$  to Alice.
- Alice applies  $d_A$  to  $(m^{e_A})^{e_B}$  and sends  $m^{e_B}$  to Bob.
- On receiving  $m^{e_B}$  Bob can recover the message using  $d_B$ .

Comparing MO and DH. Observe that MO is not a public key cryptosystem unlike DH. The base  $g$  is public in DH. Both are vulnerable to the man-in-the-middle attack unless the data is authenticated or signed. The total number of communication rounds to exchange keys is 4 and hence MO is expensive.

**Claim.** MO  $\leq$  DL

The adversary supposedly has access to  $m^{e_A}, m^{e_B}, m^{e_A e_B}$  as this data is exchanged on the network. Suppose the adversary has a black box which efficiently solves DL. On giving  $g = m^{e_A}$  and  $y = m^{e_A e_B}$  to the black box it returns  $e_B$  such that  $y = g^{e_B}$ . Now using  $e_B$  the adversary can compute  $d_B$  and hence  $(m^{e_B})^{d_B} = m$   $\diamond$

To ensure that the group generated by  $g = m^{e_A}$  is as big as possible we pick the prime  $p$  such that  $p-1 = 2q$  where  $q$  is a prime. Note that 2 factor

is unavoidable as  $p$  is odd and hence  $p - 1$  is even. In practice we should also ensure that  $p - 1$  is not  $\log p$ -smooth.

### 3. El Gamal Public Key System

We next look at a public key cryptosystem based on the DL problem over a prime finite field. Let  $g$  be the generator of the multiplicative group of  $\mathbf{F}_p$ , that is,  $\mathbf{F}_p^* = \langle g \rangle$ .

**Initialization.**

- $A$  secretly picks  $a$  and publicizes  $g^a$  as her public key.
- Similarly  $B$  picks  $b$  and publicizes  $g^b$  as his public key.

**Protocol** for Alice sending message  $m$  to Bob.

- Alice generates a random  $x$  and sends  $(g^x, m(g^b)^x)$  to Bob.
- Bob recovers  $m$  by computing  $mg^{bx}((g^b)^x)^{-1}$  (and this explains the use of the  $g^x$  sent by Alice).

The **El Gamal (EG) problem** can be formulated as follows.

Given  $g^x, mg^{xb}$  to compute  $m$ .

Can the EG problem be reduced to DL in polynomial time? Yes.

**Claim.**  $\text{EG} \leq \text{DL}$

Giving  $g^x$  to a DL-black box which computes  $x$  and then computing  $mg^{bx}((g^b)^x)^{-1} = m$ .  $\diamond$

**Claim.**  $\text{DH} \leq \text{EG}$

Giving  $g^x, g^b$  to a DH-black box which computes  $g^{bx}$  and then computing  $mg^{bx}(g^{bx})^{-1} = m$ .  $\diamond$

Comparing the encryptions functions of RSA  $E(m) = m^e$  and El Gamal  $E(x, m) = (g^x, mg^{bx})$  we realize the probabilistic nature of the latter due to the random  $x$ . Hence the EG encryption function is not a bijection between message space to itself but rather message space  $\times$  random strings to message space. Also the commutativity of encryption functions are vital to RSA and El Gamal.

The relation between Integer factoring and Discrete Logarithm is another open problem. The core idea in algorithms which solve these problems is the

*smoothness* of the group order trick. When a faster factoring is developed it is adapted to solve the DL problem quicker and vice-versa. Hence the fastest algorithms to solve the two problems have always been comparable with regards to time complexity. As long as you don't solve the problem in quadratic or cubic time the cryptosystem makers will still thrive by just doubling or tripling key sizes. In a classical computing model the two problems are not known to be related but in quantum computing model (where probabilities are allowed to be complex numbers), the same quantum algorithm solves both Integer factoring and Discrete logarithm. But quantum computers are still in their infancy being able to just factor 15 currently. Maybe if they are in their "teens" they'll be able to factor.

Arguably public key cryptography is most used for signing documents.

Nowadays you can generate 1000 bit primes quickly but cannot factor numbers of the same size.

Why is RSA the most popular system? Definitely due to *marketing* and also because Integer factoring has been studied a lot more<sup>1 2</sup> compared to the DL problem.

## 4. Digital Signature Scheme

A decryption function can be used to sign only if the scheme is deterministic like RSA for instance hence we turn away from the El Gamal scheme which is probabilistic and towards another scheme based on the DL problem.

The Digital Signature Standard (DSS) aka Digital Signature Algorithm (DSA) was published by NIST in the 1990s is a signature scheme based on public key cryptography and the DL problem. The other standard the government pushes is DES which is a symmetric key scheme.

---

<sup>1</sup>"The problem of distinguishing prime numbers from composite numbers and of resolving the latter into their prime factors is known to be one of the most important and useful in arithmetic. It has engaged the industry and wisdom of ancient and modern geometers to such an extent that it would be superfluous to discuss the problem at length. Nevertheless we must confess that all methods that have been proposed thus far are either restricted to very special cases or are so laborious and prolix that even for numbers that do not exceed the limits of tables constructed by estimated men, i.e. for numbers that do not yield to artificial methods, they try the patience of even the practiced calculator... The dignity of the science itself seems to require that every possible means be explored for the solution of a problem so elegant and so celebrated." – Karl Friedrich Gauss, *Disquisitiones Arithmeticae* (translation: A. A. Clarke) ...*was Gauss speaking Complexity Theoretically?*

<sup>2</sup>Cunningham project [2] which is to find the largest Fermat prime (which are of the form  $2^a + 1$ ) and also other types of primes.

Hence we would like it to be as short as possible but at the same time would want the group (on which the DL is presumably hard) to be as big as possible. Also entire messages are not signed but rather a hash of the message (which is substantially smaller than the message) is signed. The hash provides a trace of the message.

As usual we illustrate the scheme with a finite field with prime  $p$  elements. Moreover we pick a prime  $q$  such that  $p \equiv 1 \pmod q$ .  $\log q$  determines the size of the signature. NIST suggests that  $q \sim 160$  bits and  $p \sim 512$  bits. As usual the assumption is DL over  $\mathbf{F}_p$  is hard.

The idea being that as  $q$  is substantially smaller than  $p$ , the signature is much shorter than the public key. We know that the group  $\mathbf{F}_p^*$  is cyclic with say  $g_0$  as the generator. Hence there exists a unique (cyclic) subgroup for every divisor  $d$  of  $p - 1$  of order  $d$ , in particular of order  $q$ .

*Question.* How is the hardness of DL over  $\mathbf{F}_p$  related to the hardness of DL over a cyclic subgroup of order  $q$  where  $q|p - 1$ ?

Let's  $g = g_0^{\frac{p-1}{q}}$  be the generator of the subgroup, that is  $\mathbf{F}_q^* = \langle g \rangle$  and hence  $g^q = 1$ .

**Notation.** Say  $x \in \mathbf{F}_p$  then  $[x]$  is the reduced representative of  $x$ , that is,  $[x] = x \bmod p \in \{0, \dots, p - 1\}$ . **Ex.**  $x = 11$  and  $p = 7$ . Then  $[11] = 4$  and  $[2^4] = 2$ .

### Initialization.

- Alice selects  $a$  secretly where  $0 < a < q$  and publicizes  $y = g^a$  as her public key.

**Protocol** for Alice to sign hash  $h$  of message  $m$ .

- Alice picks random  $k$ ,  $0 < k < q$ .
- $r \leftarrow [g^k] \bmod q$ .
- Next she computes  $s$  such that  $sk \equiv h + ar \pmod q$ .
- $(r, s)$  is the signature for  $h$ .

Say Alice sends Bob message  $m$  in encrypted form with it's signed hash. Bob then retrieves  $m$  and does the following to verify that it was really Alice who sent the message.

**Verification.** Bob knows  $r, s, h, g^a$  but not  $a, k$  and wants to ascertain origin of the message.

- Bob computes  $\left[ (g^{s^{-1}h})(g^a)^{s^{-1}h} \right] \bmod q$ , where  $[ \ ]$  is the reduced representative.
- Next Bob checks whether the previous computation matches  $r$

For a malicious third party  $k, r$  are easy to forge but  $a$  is not and therefore he or she cannot compute the second part of the signature  $s$  efficiently. One way to crack DSS is to solve the linear congruence but this takes  $O(q)$ . Instead we'll see algorithms which solve the DL problem in  $O(\sqrt{q})$  and we can use these to crack DSS quicker (though not efficiently).

Comparing RSA and DSS. RSA signatures are roughly 512 bits long. On the other hand though DSS computation is done mod  $p$  which is about 512 bits, the signature consists of two 160 bit numbers  $r$  and  $s$ . Another advantage DSS offers is that there is only one  $p$  suitable for group communication unlike RSA where each player has to pick a different modulus  $n$ .

Consider the linear congruence  $uk \equiv v + aw \bmod q$ . Taking  $u = r, v = s, w = h$  we get the original DSS we've discussed above. Taking  $u = rh, v = s, w = 1$  we get a DSS variant. Refer to chapter 20 [S] for more such variants.

In the future if people come up with algorithms which solve DL quicker than current bests we would like to increase our  $p$ . An interesting question is the density of primes  $p$  such that  $p \equiv 1 \bmod q$  and  $q \sim \frac{1}{3}p$ .

### Bibliography:

1. <http://www.gchq.gov.uk/about/pke.html> GCHQ claiming to be the inventors of PKC
2. <http://www.cerias.purdue.edu/homes/ssw/cun/> Cunningham Project