# ElGamal public key cryptosystem

- Taher ElGamal was one of the founders of cryptosystems for internet browsers. He developed what is now known as the ElGamal cryptosystem for Netscape in the 1990's and was responsible for the internet protocol known as SSL or "secure socket layer". His system was based on discrete logarithms. Here is how this works:
- Bob chooses a finite $F_q$, a multiplicative generator $g$ and a number $a$. He computes $b = g^a$ and makes the triple $(F_q, g, b)$ public but keeps the knowledge of $a$, the logarithm, to himself.
- When we say that Bob chooses a field $F_q$, he actually chooses a particular representation of it in the form $Z_p[x]/(f)$. For instance, he could make the prime $p$ and the polynomial $f$ known publicly. Similarly he gives $g$ and $b$ in the form of a polynomial subject to whatever form he presents $F_q$.

# ElGamal public key cryptosystem, cont'd

- Alice also has to encode her message as a polynomial. If we think as we did with RSA that Alice is encoding a message $m$ as a number less than $q = p^n$ then we can convert $m$ to a polynomial as follows: write

$$m = a_0 + a_1 p + a_2 p^2 + \ldots + a_{n-1} p^{n-1}$$

  where $a_0, \ldots, a_{n-1}$ are integers between 0 and $p - 1$. This is $m$'s base $p$ representation or its $p$-adic representation.
- Now represent $m$ as the polynomial

$$a_0 + a_1 x + \ldots + a_{n-1} x^{n-1}.$$

- Now that Alice has her message as a polynomial, she picks a random integer $k$ and computes $r = g^k$ and $t = mb^k$ and sends the pair $(r, t)$ to Bob.
- How can Bob decode $m$ from $(r, t)$? Remember he knows $a$, the discrete log of $b$ in base $g$. He calculates

$$tr^{-a} = mb^k g^{-ak} = mg^{ak} g^{-ak} = m \text{ in } F_q.$$

# Diffie-Hellman key exchange

- A cryptosystem similar to the ElGamal system can be used for key exchange - a way to have two parties at a distance using the same key and to pass a key securely. This is the Diffie-Hellman system.
- Alice and Bob agree on a finite field $F_q$ and a generator $g$; they can do this publicly.
- Each of them separately (and secretly) choose integers $x$ for Alice and $y$ for Bob.
- Alice sends Bob $g^x$ and Bob sends Alice $g^y$.
- They each then compute $g^{xy}$ and use this as their key.

# The computational Diffie-Hellman problem

- How secure is this system? Of course if someone can compute discrete logs then this is not secure.
- The question of whether there is an algorithm for finding $g^{xy}$ from $g^x$ and $g^y$ is known as the computational Diffie-Hellman problem.
- It is not known if this is an easier problem then solving the discrete log problem.
- It is known that solving the computational DH problem is equivalent to cracking the associated El Gamal cryptosystem.
- That is, suppose we can solve the computational DH problem for a field $F_q$ and generator $g$. Now consider the ElGamal cryptosystem $(F_q, g, b)$ and Alice's transmission $(r, t)$. We know $b = g^a$ and $r = g^k$ so we can compute $g^{ak}$.
- But $t = mg^{ak}$ so $tg^{-ak} = m$ and we have decoded the message.

# The computational Diffie-Hellman problem, cont'd

- Now suppose we want to solve the computational DH problem for a field $F_q$ and generator $g$.
- Assume we can crack the ElGamal system with $(F_q, g, b)$ for any $b$.
- Then let $b = g^x$, $r = g^y$ and $t = 1$.
- If we now decode $(r, t)$ we get $m = g^{xy}$.