

The ElGamal Public Key Encryption Algorithm

The ElGamal Algorithm provides an alternative to the RSA for public key encryption.

- 1) Security of the RSA depends on the (presumed) *difficulty of factoring large integers*.
- 2) Security of the ElGamal algorithm depends on the (presumed) *difficulty of computing discrete logs* in a large prime modulus.

ElGamal has the disadvantage that the ciphertext is twice as long as the plaintext.

Alice chooses

- i) A large prime p_A (say 200 digits),
- ii) A primitive element α_A modulo p_A ,
- iii) A (possibly random) integer d_A with $2 \leq d_A \leq p_A - 2$.

Alice computes

iv) $\beta_A \equiv \alpha_A^{d_A} \pmod{p_A}$.

Alice's *public key* is (p_A, α_A, β_A) . Her *private key* is d_A .

Bob encrypts a short message M ($M < p_A$) and sends it to Alice like this:

- i) Bob chooses a random integer k (which he keeps secret).
- ii) Bob computes $r \equiv \alpha_A^k \pmod{p_A}$ and $t \equiv \beta_A^k M \pmod{p_A}$, and then discards k .

Bob sends his encrypted message (r, t) to Alice.

When Alice receives the encrypted message (r, t) , she decrypts (using her private key d_A) by computing tr^{-d_A} .

$$\begin{aligned} \text{Note } tr^{-d_A} &\equiv \beta_A^k M (\alpha_A^k)^{-d_A} \pmod{p_A} \\ &\equiv (\alpha_A^{d_A})^k M (\alpha_A^k)^{-d_A} \pmod{p_A} \\ &\equiv M \pmod{p_A} \end{aligned}$$

Even if Eve intercepts the ciphertext (r, t) , she cannot perform the calculation above because she doesn't know d_A .

$$\beta_A \equiv \alpha_A^{d_A} \pmod{p_A}, \text{ so } d_A \equiv \log_{\alpha_A}(\beta_A) \pmod{p_A}.$$

Eve can find d_A if she can compute a discrete log in the large prime modulus p_A , presumably a computation that is too difficult to be practical.