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.

It has the advantage the same plaintext gives a different ciphertext (with near certainty) each time it is encrypted.

Alice chooses

i) A large prime $\boldsymbol{p_A}$ (say 200 to 300 digits),

ii) A primitive element $\boldsymbol{\alpha_A}$ modulo p_A,

iii) A (possibly random) integer $\boldsymbol{d_A}$ with $2 \le d_A \le p_A - 2$.

Alice computes

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

Alice's *public key* is $(\boldsymbol{p_A}, \boldsymbol{\alpha_A}, \boldsymbol{\beta_A})$. Her *private key* is $\boldsymbol{d_A}$.

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 $\boldsymbol{r \equiv \alpha_A^{k} \pmod{p_A}}$ and $\boldsymbol{t \equiv \beta_A^{k} M} \pmod{p_A}$, and then discards k.

Bob sends his encrypted message $\boldsymbol{(r, t)}$ to Alice.

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

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

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

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

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.

Caution: Bob should choose a different random integer k for each message he sends to Alice.

If M is a longer message, so it is divided into blocks, he should choose a different k for each block.

Say he encrypts two messages (or blocks) M_1 and M_2, using the same k, producing ciphertexts

$$(r_1, t_1) = (\alpha_A{}^k, \beta_A{}^k M_1), \quad (r_2, t_2) = (\alpha_A{}^k, \beta_A{}^k M_2).$$

Then $t_2 t_1^{-1} \equiv M_2 M_1^{-1} \pmod{p}$, $M_2 \equiv t_2 t_1^{-1} M_1 \pmod{p}$. If Eve intercepts both ciphertext messages and discovers one plaintext message M_1, she can compute the other plaintext message M_2.

Example: Alice chooses $p_A = 107$, $\alpha_A = 2$, $d_A = 67$, and she computes $\beta_A = 2^{67} \equiv 94 \pmod{107}$. Her public key is $(p_A, \alpha_A, \beta_A) = (2,67,94)$, and her private key is $d_A = 67$.

Bob wants to send the message "B" (66 in ASCII) to Alice. He chooses a random integer $k = 45$ and encrypts $M = 66$ as $(r, t) = (\alpha_A{}^k, \beta_A{}^k M) \equiv (2^{45}, 94^{45} 66) \equiv \mathbf{(28, 9)} \pmod{107}$. He sends the encrypted message (28, 9) to Alice.

Alice receives the message $(r, t) = (28, 9)$, and using her private key $d_A = 67$ she decrypts to

$$t r^{-d_A} = 9 \cdot 28^{-67} \equiv 9 \cdot 28^{106-67} \equiv 9 \cdot 43 \equiv \mathbf{66} \pmod{107}.$$