Review. ElGamal encryption

- Like RSA, ElGamal is terribly slow compared with symmetric ciphers like AES.

  Encryption under ElGamal requires two exponentiations (slower than RSA); however, these exponentiations are independent of the message and can be computed ahead of time if need be (in that case, encryption is just a multiplication, which is much faster than RSA). Decryption only requires one exponentiation (like RSA).

- In contrast to RSA, ElGamal is randomized. That is, a single plaintext m can be encrypted to many different ciphertexts.

  A drawback is that the ciphertext is twice as large as the plaintext.

  On the positive side, an attacker who might be able to guess potential plaintexts cannot (as in the case of vanilla RSA) encrypt these herself and compare with the intercepted ciphertext.

Example 169. If Bob selects $p = 23$ for ElGamal, how many possible choices does he have for g? Which are these?

Solution. g needs to be a primitive root modulo 23. Recall that, modulo a prime p, there are $\phi(\phi(p)) = \phi(p-1)$ many primitive roots. Hence, Bob has $\phi(p-1) = \phi(22) = 10$ choices for g.

Example 170. Does Alice have to choose a new y if she sends several messages to Bob using ElGamal encryption?

Solution. Yes, she absolutely has to randomly choose a new y every time! Here's why:

If she was using the same y to encrypt messages $m^{(1)}$ and $m^{(2)}$, Alice would be sending the ciphertexts $(c_1^{(1)}, c_2^{(1)}) = (g^y, g^{xy} m^{(1)})$ and $(c_1^{(2)}, c_2^{(2)}) = (g^y, g^{xy} m^{(2)})$.

That means, Eve can immediately figure out $c_2^{(1)} / c_2^{(2)} = m^{(1)} / m^{(2)}$ (the divison is a modular inverse and everything is modulo p). That's a combination of the plaintexts, and Eve should never be able to get her hands on such a thing.

(Note that Eve would know right away if Alice is doing the mistake of reusing y because $c_1^{(1)} = c_1^{(2)}$.)

Comment. The situation is just like for the one-time pad (in that case, reusing the key reveals $m^{(1)} \oplus m^{(2)}$).

The computational and decisional Diffie–Hellman problem

We indicated that the security of ElGamal depends on the difficulty of computing discrete logarithms. Here is a more precise statement.

Theorem 171. Decrypting c to m in ElGamal is exactly as difficult as the **computational Diffie–Hellman problem** (CDH).

The CDH problem is the following: given $g, g^x, g^y \pmod p$, find $g^{xy} \pmod p$. It is believed to be hard.

Proof. Recall that the public key is $(p, g, h) = (p, g, g^x)$. The ciphertext is $c = (g^y, h^y m) = (g^y, g^{xy} m)$.

Hence, determining m is equivalent to finding g^{xy}.

Since $g, g^x, g^y \pmod p$ are known, this is precisely the CDH problem. □

Armin Straub
straub@southalabama.edu

Example 172. In fact, even the **decisional Diffie–Hellman problem** (DDH) is believed to be difficult.

The DDH problem is the following: given g, g^x, g^y, $r \pmod p$, decide whether $r \equiv g^{xy} \pmod p$. Obviously, this is simpler than the CDH problem, where g^{xy} needs to be computed. Yet, it, too, is believed to be hard.

Comment. Well, at least it is hard (modulo p) if we always want to do better than guessing.

Here's how we can sometimes do better than guessing: if g^x or g^y are quadratic residues (this is actually easy to check modulo primes p using quadratic reciprocity and the Legendre symbol), then g^{xy} is a quadratic residue (why?!). Hence, if r is not a quadratic residue, we can conclude that $r \not\equiv g^{xy}$.

More on safe primes

Recall that p is a **safe prime** if both p and $(p-1)/2$ are prime. The next example illustrates why it is common to use safe primes for ElGamal.

In general, it is difficult to ensure that g is a primitive root, or almost a primitive root, modulo p.

Example 173. Suppose that p is a safe prime. Show that all residues $g \not\equiv 0, \pm 1 \pmod p$ have order $(p-1)/2$ or $p-1$.

In the latter case, g is a primitive root. In fact, if $p > 5$, then half of the residues $g \not\equiv 0, \pm 1$ are primitive roots.

Solution. Suppose $g \not\equiv 0, \pm 1 \pmod p$. Because p is a prime and $g \not\equiv 0$, g is invertible. Its multiplicative order N divides $\phi(p) = p-1$. But the prime factorization of $p-1$ is 2 times $(p-1)/2$. Hence, the only possible orders are $1, 2, (p-1)/2$ and $p-1$. The residues ± 1 are the only with order 1 and 2 (why?!). Thus, g must have order $(p-1)/2$ or $p-1$.

Finally, if $p > 5$ (so that $(p-1)/2$ is odd), note that the number of primitive roots is $\phi(p-1) = \phi(2)\phi((p-1)/2) = (p-3)/2$, which is exactly half of the residues g.

Advanced comment. Actually, it is easy to distinguish between the residues that have order $(p-1)/2$ and those that have order $p-1$. Recall that, if x has order $p-1$, then x^2 has order $\frac{p-1}{\gcd(p-1,2)} = \frac{p-1}{2}$. It follows that (among the $x \not\equiv 0, \pm 1$) quadratic residues have order $(p-1)/2$. (And, using quadratic reciprocity, it is computationally easy to determine whether a residue modulo p is a quadratic residue or not.)

Example 174. Is there any advantage for RSA if p is a safe prime? Potential issues?

Solution. If p is a safe prime, then $\gcd(p-1, q-1) = 2$. Why?!

Hence, the key space is as large as possible.

On the other hand, we need to think about whether we are weakening the security in case we might severely limit the number of possible p's to choose from.

Another issue is that generating random safe primes is considerably more work. On the other hand, Bob usually does not generate a public key frequently, so that this might not be much of an issue.