

Symmetric Re-encryption

James Heather
University of Surrey

◇ Re-encryption ◇

We have some encryption

$$E_k(r, m)$$

from which we can recover m (and maybe r) with k^{-1} . Re-encryption involves turning it into

$$E_k(r', m)$$

without knowledge of k^{-1}, r, m .

Usually this is in an asymmetric setting: ElGamal, Paillier, etc.

◇ Motivation ◇

One key motivating area is e-voting. We take a batch of encrypted votes, re-encrypt each, and mix them up:

$$\{E_k(r_1, v_1), \dots, E_k(r_n, v_n)\} \longrightarrow \{E_k(r'_1, v_1), \dots, E_k(r'_n, v_n)\}$$

so that the set of plaintexts remains the same but no-one can perform end-to-end matching.

This is fine as long as the plaintexts stay small. But if they get very big, the asymmetric crypto starts to look unwieldy.

◇ Hybrid encryption ◇

What do you do if you want to encrypt a very long plaintext (say, a DVD) with your public key?

$$E_{PK}(r, k). \{m\}_k \quad \text{or} \quad E_{PK}(r, k). \{IV, m\}_k$$

This requires a very small bit of asymmetric crypto, and lots of (fast) symmetric crypto. What if you want to apply re-encryption to this?

◇ Two options ◇

There are two options for what to do with this to re-encrypt it. (Obviously it will not do just to re-encrypt the public part.)

Option 1: change k :

$$E_{PK}(r', k').\{m\}_{k'}$$

Option 2: leave k unchanged, but change some randomness in the symmetric part:

$$E_{PK}(r', k).\{IV, m\}_k$$

◇ Leaving k unchanged ◇

Here, dealing with the asymmetric part is just standard re-encryption.

It seems that we would need a block cipher. But it will have to be *malleable*.

E.g., in DES:

$$\overline{\{m\}_k} = \{\overline{m}\}_{\overline{k}}$$

But even if we could find an appropriate cipher, this wouldn't work, at least for e-voting.

◇ End-to-end matching ◇

The problem here is that if the decryptions are eventually published, we can match end-to-end:

$$E_{PK}(r, k). \{IV, m\}_k \longrightarrow \dots \longrightarrow E_{PK}(r', k). \{IV', m\}_k$$

At the far end, if k is eventually published (after mixing), we can try all of the original ciphertexts to see which one decrypts to m using k .

So we need to change k .

◇ Changing k ◇

Now we have two problems: changing the symmetric part, and changing the asymmetric part.

$$E_{PK}(r, k). \{m\}_k \longrightarrow E_{PK}(r', k'). \{m\}_{k'}$$

We need a function that can be applied to k :

1. *inside* the asymmetric part;
2. inside the *key* of the symmetric part

without knowing SK , k or m .

◇ Exclusive-or ◇

The best bet is probably exclusive-or.

Sticking points:

1. For the asymmetric part, we are working modulo p , p^2 , pq , or some such. What does exclusive-or even *mean* here?
2. Again, we need some malleability in the symmetric part, and usually symmetric ciphers are non-malleable by design.

◇ Asymmetric: Goldwasser-Micali ◇

Goldwasser-Micali (based on quadratic residue problem) is used to encrypt individual *bits*. It has an interesting homomorphic property:

$$E(b_0) \oplus E(b_1) = E(b_0 \oplus b_1)$$

This means that we can encrypt k as a sequence of encrypted bits $\langle E(k_i) \rangle$. Now we turn this into an encryption of $k \oplus x$ by turning it into $\langle E(k_i) \oplus E(x_i) \rangle$.

This works, but inflates the size of the asymmetric part. Whether that is acceptable depends on the size of m .

◇ Symmetric: Linear Feedback Shift Register ◇

A linear feedback shift register has the property that

$$S(k) \oplus S(k') = S(k \oplus k')$$

Since the symmetric part is encrypted as $m \oplus S(k)$, we just encrypt again using x :

$$\{\{m\}_k\}_x = m \oplus S(k) \oplus S(x) = \{m\}_{k \oplus x}$$

But LFSRs are not strong enough for good security for most applications (e.g., KPA).

◇ Conclusions ◇

We have made some progress, but not much: security is weak, and efficiency is dubious.

Maybe this is impossible without a specially designed malleable cipher?

Is there a call for homomorphic symmetric encryption?