An Improved Construction for Universal Re-encryption.

Peter Fairbrother

10 Sheepcote Barton
Trowbridge BA14 7SY UK
peter@m-o-o-t.org

Abstract. Golle et al recently introduced universal re-encryption, defining it as re-encryption by a player who does not know the key used for the original encryption, but which still allows an intended player to recover the plaintext. Universal re-encryption is potentially useful as part of many information-hiding techniques, as it allows any player to make ciphertext unidentifiable without knowing the key used.

Golle et al's techniques for universal re-encryption are reviewed, and an improved hybrid universal re-encryption construction which permits indefinite re-encryptions with better efficiency and an almost-optimally small increase in file size is presented. Some implementational issues and optimisations are discussed.

1 Introduction

Golle et al [1] recently introduced universal re-encryption, defining it as re-encryption by a player who does not know the key used for the original encryption, but which still allows an intended player to recover the plaintext.

Golle et al proposed universal re-encryption for use in mixnets and anonymous bulletin boards, but it has many other potential uses in anonymous communications and untraceable messaging. It can also be used to provide cover against observation-based attacks on the plausible deniability of steganographic filing systems [2]. It is generally a useful addition to the toolkit of information hiding technologies.

The constructions of Golle et al, while secure and effective, have some important drawbacks. Their simple system increases the ciphertext to four times the plaintext size, and is computationally very expensive, requiring two El Gamal encryptions with the four associated large modular exponentiations per block. This can be improved for large files, but it remains expensive both computationally and in terms of ciphertext size.

Their hybrid system has the disadvantages of taking significant effort and space in managing the keys, often more effort than their simple system, but more importantly of limiting the number of re-encryptions possible before the plaintext becomes unrecoverable; further, it leaks information about the history of the underlying plaintext, making the system ineffective or unuseable in many situations.

The construction presented here reduces the computational requirements to a single modular exponentiation per block, with no increase in file size (excepting a single key storage block per file). It imposes no limit to the number of possible re-encryptions, and no historical information is leaked, so it can be used in almost every situation.

2 Universal Cryptosystems

2.1 Golle et al's simple system.

In Golle et al's simple system plaintext is encrypted as a pair of El Gamal ciphertexts. The first is a standard encryption of the plaintext to an El Gamal public key, the second is an El Gamal encryption of unity to the same public key - a "unit" - but encrypted with a different random x.

A "unit" has the following properties, as far as the holder of the relevant secret key is concerned: any player can generate new "units" from a known "unit", without knowing the public key used to generate the "unit". A part-by-part multiplication of an El Gamal ciphertext by a "unit" encrypted with the same public key does not change the underlying plaintext.

Without knowledge of the secret keypart, it is presumed (under the Decisional Diffie-Hellman assumption) to be difficult to identify a ciphertext multiplied by an unknown "unit" with the original ciphertext. It is also presumed to be difficult to identify a generated "unit" as being generated from another "unit".

In Golle et al's system, on re-encryption two new "units" are generated from the original "unit"; the player part-wise multiplies one with the standard ciphertext, the second replaces the "unit" ciphertext in the exposed ciphertext pair. The original "unit" is discarded.

The holder of a secret keypart can ascertain whether the constructed pair is encrypted to it by decrypting the second ciphertext using his secret keypart. If the decrypted value is one, he can then decrypt the re-encrypted ciphertext; else the ciphertext is not encrypted to his public key.

A player who does not have the El Gamal secret keypart cannot identify a ciphertext as a "unit" encrypted to a specific key; and he cannot identify a ciphertext multiplied by an unknown "unit" with the unmultiplied ciphertext.

Problems A pair of El Gamal ciphertexts is four times the size of the corresponding plaintext file, requiring four times the computation, storage and transport of ordinary, non-universal, cryptosystems.

Each pair will take 4×k-bit data blocks to store a single k-bit block of information, where k is the first security parameter, the size of the El Gamal modulus. k will be at least 1024 bits for present-day security. Files will typically be many times that size, and split into blocks.

A small improvement on Golle's construction, for larger files, reducing the size requirements to approaching twice file size, would be splitting the file into chunks and encrypting them as simple El Gamal ciphertexts with different "units", concatenating, and appending only one "unit" to the whole. On re-encryption as many "units" as needed can be generated from the single "unit" and used to camouflage the individual blocks. This does not change the overall workload however.

- Key Generation: Output is an El Gamal keypair $x, y\ (= g^x)$p. p and g are usually used and held in common.

- Encryption: Input is a file F of f k-bit blocks; an El Gamal public key (y, g, p); and random $k_1 \ldots k_f, k_u \in$ Zp

Output is a ciphertext $C = [(\alpha_1, \beta_1); (\alpha_2, \beta_2) \ldots (\alpha_f, \beta_f)]; [(\alpha_u, \beta_u)]$
$= [(F_1 y^{k_1}, g^{k_1}); (F_2 y^{k_2}, g^{k_2}) \ldots (F_f y^{k_f}, g^{k_f})]; [(y^{k_u}, g^{k_u})]$.

- Re-encryption: Input is a ciphertext C; and random $k'_1 \ldots k'_f, k'_u \in$ Zp.

Output is a ciphertext $C' = [(\alpha'_1, \beta'_1); (\alpha'_2, \beta'_2) \ldots (\alpha'_f, \beta'_f)]; [(\alpha'_u.\beta'_u)]$
$= [(\alpha_1 \alpha_u^{k'_1}, \beta_1 \beta_u^{k'_1}); (\alpha_2 \alpha_u^{k'_2}, \beta_2 \beta_u^{k'_2}) \ldots (\alpha_f \alpha_u^{k'_f}, \beta_f \beta_u^{k'_f})]; [(\alpha_u^{k'_u}.\beta_u^{k'_u})]$.

- Decryption: Input is a ciphertext C (or C'); and a private key x.

If $\alpha_u / \beta_u^x = 1$ then the output is $[(\alpha_1/\beta_1^x); (\alpha_1/\beta_1^x) \ldots (\alpha_1/\beta_1^x)]$. If not, the file is not decryptable by (or meant for) the holder of that private key.

The computational requirements however remain at the same very high level, and the ciphertext size is still over twice file size (2M + (4×k)); while this is an improvement, it is still very costly overall.

2.2 Golle et al's hybrid system

Golle et al also proposed a hybrid universal cryptosystem, where the file is conventionally encrypted with a symmetric cipher and the key is appended, stored using a simple universal cryptosystem. To allow re-encryption extra "blank" re-encryptable keys are appended. On re-encryption the cipherterxt is re-encrypted with the conventional cipher, using a random key which is then stored in one of the "blank" keys, and the position of the "blank" keys is rotated.

Problems This adds 4×k bits per "blank" key to the file; in mixnet use, if a typical 4kB message is considered, with 1024-bit security parameter k, then only eight "blank" keys will double message size. A more realistic number of "blank" keys will make this system actually worse in terms of traffic requirements than a simple system.

Furthermore, the "blank" keys must be re-encrypted or decrypted as appropriate. In many situations, including the mixnet situation above, the hybrid system will also end up needing more work than a simple system.

The number of re-encryptions a message has undergone is available to any observer, and this significantly impacts the untraceability properties of the system. In a mixnet, only messages with the same number of re-encryptions can be grouped for anonymity. This is a very significant drawback, especially considering the size of present mixnets.

Possibly the worst drawback of this hybrid system however is that only a limited number of re-encryptions can be performed before the plaintext becomes unrecoverable. This significantly affects the useability of this construction.

Golle et al suggest that in mixnet situations when all the blank re-encryption keys have been used the message should be withdrawn and archived - but in a mixnet, and indeed in any situation where universal re-encryption is likely to be useful, no information about the number of re-encryptions should be available to anyone, as it would leak significant information to an observer about which input ciphertexts relate to which output ciphertexts.

If no information about the number of re-encryptions is available it is impossible to do this archiving, and thus to use this construction; or if some information is available at best it leaks information useful for identifying re-encrypted ciphertexts, defeating the purpose of the system. In many cases that will make this hybrid construction ineffective or unuseable.

3 An Improved Hybrid Construction.

We present this improved construction. It is similar to Golle et al's hybrid construction, but the file is encrypted using the Pohlig-Hellman secret key algorithm [3], and only a single key-block is appended to hold the secret key.

Pohlig-Hellman is chosen as the symmetric algorithm because on re-encryption a player can, using the group multiplicative properties of Pohlig-Hellman keys, create a new "overall" key - and using similar multiplicative properties of El Gamal keys (properties preserved in a simple universal cryptosystem based on it), a player can calculate and store this new "overall" symmetric key in the single key-block without knowing (or being able to calculate) either the initial key, the calculated "overall" key, or the public key used to store it in the key-block.

Pohlig-Hellman, as used here, works like this:

- Encryption: $C = M^e$ mod p
- Re-encryption: $C' = C^{e'}$ mod p
- Decryption: First find d, the inverse of ee'e"... mod p, such that $de = 1$ mod (p-1): then $M = C^d$ mod p.

(C - ciphertext: M - message)

A k-bit "safe" (= 2q+1, q prime) prime is chosen for p. p is not secret, and would normally be shared by all players in a system.

Generating suitable Pohlig-Hellman keys is simply a matter of concatenating a random number of suitable size with 1 in order to ensure the key is odd, and thus that a unique inverse of the "overall" key exists mod phi(p). This is essential for decryption. A solution to $de = 1$ mod (p-1) exists only if $(e, p-1)|1$. p-1 is even for prime p, so e must be odd. q should not be used as a key, but the probability of that happening if the key is generated this way is so low that we ignore it.

The encryption key e is stored in a simple universal re-encryption block appended to the ciphertext; but we use q, not p, as the El Gamal modulus. On re-encryption we generate at random a new key e', and exponentiate the main ciphertext to that value. We also multiply the first part of the first of the El Gamal ciphertext pairs in the universal block by e', modulo q. Then we do a simple re-encryption of the El Gamal universal block, again modulo q. The value stored in the universal block is now ee' mod q.

If we know the secret El Gamal keypart, to find the relevant Pohlig-Hellman decryption key d we need to know the "overall" encryption key, which is equal to ee' mod (p-1), $= ee'$ mod (2q).

ee' mod q is the value stored in the universal key block, and we wish to find ee' mod (2q). As all the intermediate keys are odd the resultant "overall" key must also be odd, so if the result of decrypting the universal block is even adding q will give the "overall" key, otherwise the result is the "overall" key. We find the modular inverse of this value, and use it to decrypt the main file.

- Setup: Output is p, q ($= (p$-$1)/2$), g; such that p, q are prime, and g is a generator of q (or of a subgroup of q; q may be of special form, see below). p and g are usually used and held in common.

- Key Generation: Output is an El Gamal keypair x, y ($= g^x$ mod q).

- Encryption: Input is a file F of f k-bit blocks; an El Gamal public key (y, g, p); a random Pohlig-Hellman key $e \in_u \mathrm{Z}p$, e mod $2 = 1$; and random k_0, $k_u \in \mathrm{Z}q$.

Output is a ciphertext C = $[\psi_1\ ,\psi_2 \ldots \psi_f];[(\alpha_0, \beta_0); (\alpha_u, \beta_u)]$

$= [F_1^e \bmod p\ ,F_2^e \bmod p \ldots F_f^e \bmod p];[(ey^{k_0} \bmod q,\ g^{k_0} \bmod q);(y^{k_u} \bmod q,\ g^{k_u} \bmod q)]$.

- Re-encryption: Input is a ciphertext C (or C'); a random $e' \in \mathrm{Z}p$, e' mod $2 = 1$; and random k_0', $k_u' \in \mathrm{Z}q$.

Output is a ciphertext $\mathrm{C}' = [\psi_1'\ ,\psi_2' \ldots \psi_f'];[(\alpha_0', \beta_0'); (\alpha_u', \beta_u')]$

$= [\psi_1^{e'} \bmod p\ ,\psi_2^{e'} \bmod p \ldots \psi_f^{e'} \bmod p];[(e'\alpha_0\alpha_u^{k_0'} \bmod q,\ \beta_0\beta_u^{k_0'} \bmod q);(\alpha_u^{k_u'} \bmod q,\ \beta_u^{k_u'} \bmod q\ )]$.

- Decryption: Input is a ciphertext C (or C') $= [\psi_1\ ,\psi_2 \ldots \psi_f];[(\alpha_0, \beta_0); (\alpha_u, \beta_u)]$; and a secret key x.

If $\alpha_u/\ \beta_u^x$ mod $q = 1$ then calculate $E = (\alpha_1/\beta_1^x \bmod q)$; iff E even, $E = E + q$; find d, the inverse mod p of E.
Output is a file F = $(\psi_1^d \bmod p;\ \psi_2^d \bmod p;\ \ldots \psi_f^d \bmod p)$.
If $\alpha_u/\ \beta_u^x$ mod $q \neq 1$, the file is not decryptable by (or meant for) the holder of that private key.

This construction is reasonably computationally efficient, increases the ciphertext by only 4×k bits, permits unlimited re-encryptions, and gives no information about the number of re-encryptions undergone, greatly increasing useability.

4 Some Implementation issues

4.1 Security

The security parameter, k, is the size in bits of the primes used for the Pohlig-Hellman (k-1 bit primes for the El Gamal) moduli. k should be chosen so that finding discreet logarithms and DHP is hard. Typical minimum values are 1,024 bits.

4.2 Encryption mode

The Pohlig-Hellman cipher is presented here in what is effectively ECB mode, with all the malleability and other problems that that mode can have. These can easily be overcome with standard modes and techniques - one solution is to use OAEP [5] or it's variants on the plaintext. A fast solution, as used by the author in his online SFS work, is to pre-encrypt the plaintext using a symmetric cipher in CBC mode with a random IV.

4.3 Speed

Pohlig-Hellman is much slower than a modern symmetric cipher, but with modern processors and typical bandwidths this need not be a problem. A not-highly-optimised implementation re-encrypts 1024-bit blocks with 160-bit exponents at 512 kb/s using 60-65% cpu utilisation on an Athlon 2600+.

4.4 Optimisations

There is no useful subgroup of prime order in the Pohlig-Hellman group, which means that decryption will involve exponentiation to a full k-bit exponent. However we assume that decryption will only be done once, by the recipient, and that re-encryption will be the most common operation. This is undoubtably true in a mixnet situation, and likely to be the case in all situations where re-encryption is useful.

A full-size exponentiation is not necessary for security of re-encryption. For the 1024 bit k example an exponentiation key of about 160 bits is sufficient [4].

It is eminently possible to have a subgroup of prime order in the field used for the El Gamal universal key-storage block, and this is desirable in order to speed up a potential recipient's identification of which ciphertexts are meant for him. Using a subgroup of prime order around 160 bits will give an order of magnitude performance improvement here without impact on overall security.

4.5 Semantic Security

Semantic security of the re-encryption block For general security it is essential that the order of g in Z_q be large and prime. 160 bits would be a typical size when used with 1024-bit k. As an attacker can tell whether or not a candidate block is a member of the subgroup, for semantic security it is essential that only members of that subgroup be accepted - for instance the attacker may be able to insert blocks that are not members of that subgroup and identify them after re-encryption. It is normally essential that a re-encryptor tests all blocks that are presented to him for membership of the relevant subgroup.

As the re-encrypter is going to re-encrypt the presented block unless it fails the test, and as most blocks can be expected to pass the test, it is convenient to combine the testing and re-encryption. The first 160 intermediate modular squarings will be calculated in order to do the re-encrypting modular exponentiation, and these values can also be used to calculate the value of the presented value exponentiated to the order of the subgroup. This will equal 1 iff the presented value is a member of the subgroup.

The order of the subgroup could be chosen with a low Hamming weight in order to speed up these checks. We believe that a low Hamming weight would not affect overall security, as it does not make any of the well-known attacks easier, but we would like to see more analysis before recommending it.

Semantic security of the Pohlig-Hellman blocks A similar limitation applies to the plaintext encrypted in the Pohlig-Hellman blocks, but here the order of the subgroup is q. Again the re-encrypter must check that every value he is presented with is a member of the subgroup, and reject any that are not.

A full-length exponentiation would be required to calculate the presented value exponentiated to the order of the subgroup, so the method above is not so attractive when the re-encryption exponentiation is limited in size for speed. However as p = 2q+1 the subgroup of order q is identical to the subgroup of quadratic residues, and the "Euclidean-like" test [6] for quadratic residuosity can be used to advantage.

The remaining problem is to ensure the plaintext values initially encrypted are quadratic residues mod p (QR). One method is as follows : first, choose q so that q = 1 mod 4. Thus p = 3 mod 8, and (2/p) = -1 (ie 2 is a non-quadratic residue mod p, or NQR). Prepend the bits 001 to the plaintext block, and test the result for QR. If it is a QR then pass to the encrypter, if not a QR then shift left before doing so. The latter has the effect of multiplying the appended plaintext by 2, and changing it into a QR (a NQR multiplied by a NQR, 2 in this case, is a QR).

On decryption, shift right if the second bit of the block is 1. Discard the first three bits.

Because the Pohlig-Hellman exponents are odd and != q it is also possible to use the set of NQR's instead of the group of QR's, but as that set is not as well studied we do not recommend it.

4.6 Future directions

While it would be computationally advantageous to replace Pohlig-Hellman with a faster symmetric cipher, no suitable well-reviewed secure fast cipher with the required properties exists, and developing one is a formidable task. Such a cipher would of necessity be a group, requiring a doubled keysize because of possible birthday and cycling attacks [7] based on the group property.

Such a cipher would potentially have other uses, in Atomic Proxy Cryptography [8], trusted parties, general re-encryption, and elsewhere, so perhaps one may be developed.

5 Conclusions

The hybrid construction presented here improves on the previous constructions, being almost optimal in size requirements and considerably more efficient computationally. The removal of leakable information and limitations on the number of possible re-encryptions also makes it far more widely useful than the previous hybrid system.

Acknowledgements: Discussions with Ari Juels, Debra Cook, Paul Rubin and John Malley have improved the paper, and made me aware of other research and the accepted names for some wheels I reinvented. I thank them for their time and kindness.

References:

[1]P. Golle, M. Jakobsson, A. Juels and P. Syverson, Universal Re-encryption for Mixnets. RSA-CT 2004 http://citeseer.nj.nec.com/golle02universal.html

[2] Peter Fairbrother, Observation-Based Attacks on Steganographic File Systems. Forthcoming

[3] US Patent 4,424,414 (Expired)

[4] NIST. Special Publication 800-57: Recommendation for Key Management. Part 1: General Guideline. Draft, January 2003.

[5] M. Bellare and P. Rogaway, Optimal asymmetric encryption, Advances in Cryptology - Eurocrypt '94, Springer-Verlag (1994), 92-111.

[6] eg V. Shoup, A Computational Introduction to Number Theory and Algebra beta3, ch.13 s.1 p274 http://shoup.net/ntb/ntb-b3.pdf

[7] Burton S. Kaliski Jr,. Ronald L. Rivest, and Alan T. Sherman, Is the Data Encryption Standard a Group? Eurocrypt '85

[8] M. Blaze and M. Strauss, Atomic Proxy Cryptography, Technical report 98.5.1, AT&T research laboratories, http://citeseer.nj.nec.com/blaze98atomic.html