Improved Constructions for Universal Re-encryption.

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

November 26, 2003

Abstract

Golle et al [1] 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 improved constructions for both simple and hybrid universal re-encryption systems are presented, including a hybrid construction that permits indefinite re-encryptions with better efficiency and an almost-optimally small increase in file size. Some implementational issues and possible future directions are also discussed.

1 Introduction

Golle et al [1] 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 originally proposed universal re-encryption for use in mixnets and for anonymous bulletin boards, though it can be used for many other purposes. It can be used to provide cover against observation-based attacks on the plausible deniability of steganographic filing systems [2], and it has potential uses in anonymous communications and untraceable messaging. It is a useful addition to the toolkit for creating information hiding technologies.

The constructions of Golle et al, while secure and effective, have some 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.

Their hybrid system has the disadvantage of needing significant effort and space in managing the keys, but more importantly of limiting the number of re-encryptions possible before the plaintext becomes unrecoverable - which makes the system unuseable in many situations.

The constructions presented here reduce those requirements, and especially the hybrid construction reduces the computational requirement to a single modular exponentiation per block, with a size-limited exponent further decreasing workload, and no increase in file size, excepting a small overhead for key storage. There is no limit to the number of possible re-encryptions, so it can be used in almost every situation.

2 Simple 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 ciphertext is a standard ciphertext to an El Gamal public key, the second is an encryption of unity, a "unit" as far as a player who holds the corresponding secret El Gamal key is concerned.

A "unit" has the following properties - if another El Gamal ciphertext encrypted with the same public key is part-by-part multiplied by a "unit" it does not change the underlying plantext. Any player can generate new "units" from an existing "unit", without knowing the key used to generate the "unit". It is difficult to establish whether a ciphertext is a "unit" derived from another "unit" without knowledge of the secret keypart.

In Golle et al, on re-encryption two new "units" are generated from the original "unit"; one is part-wise multiplied with the standard ciphertext, and the second replaces the "unit" ciphertext in the 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, and checking whether the decrypted value is one. He can then decrypt the re-encrypted ciphertext.

A player who does not have the El Gamal secret keypart cannot identify a ciphertext as a "unit", and cannot identify a ciphertext multiplied by an unknown "unit" with the unmultiplied ciphertext.

2.2 A small improvement.

A pair of El Gamal ciphertexts is four times the size of the correspnding plaintext file, requiring four times the storage and transport of other, non-universal, cryptosystems.

Each pair will take $4\times$ k-bit data blocks to store a k-bit block of information, where k is the first security parameter, the size of the El Gamal modulus. k will

be around 1024 bits for present-day security. Files will typically be many times that size, and split into blocks.

The size requirements of larger files can however be reduced to approaching twice file size by splitting the filee into chunks and encrypting them as simple El Gamal ciphertexts, concatenating, and appending only one "unit" to the whole. On re-encryption as many "units" as are needed can be generated from the single "unit" and used to camouflage the individual blocks.

- 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 $k_1 \dots k_f, k_u \in$ Zp

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

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

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

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

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

The computational requirements however remain very high, and the ciphertext size is still over twice file size (2M + (4×k)); while an improvement, this is still a disadvantage.

3 Hybrid Universal Cryptosystems

3.1 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 m extra "blank" re-encryptable keys are also appended, and their position is rotated on re-encryption.

This adds 4×m×k bits to the file; but the worst drawback of the system is that only m re-encryptions can be performed before the plaintext becomes unrecoverable. This significantly affects the usefulness of this construction.

Golle et al suggest that in mixnet situations ciphertexts which have been re-encrypted m times 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 other than the intended recipient, as it would leak significant information to a player or observer about which input ciphertexts relate to which output ciphertexts.

If no information about the number of re-encryptions is available then it is impossible to do this archiving; 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 unuseable.

3.2 An Improved Hybrid Construction.

This improved construction is very similar to the above hybrid construction, but the file is encrypted using the Pohlig-Hellman secret key algorithm [3] with a single simple universally-re-encryptable system block appended, used to hold the secret key.

Pohlig-Hellman is chosen as the symmetric algorithm because on re-encryption we can, by using the associative property of Pohlig-Hellamn, generate a new equivalent "overall" key - and by using the multiplicative properties of Pohlig-Hellman keys, and the similar multiplicative properties of El Gamal (properties preserved in the simple universal cryptosystem based on it), generate and store this new "overall" symmetric key in the universal block, without knowing either the original key or the new "overall" key.

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

- Encryption: $C = M^e \bmod p$

- Re-encryption: $C' = C^{e'} \bmod p$

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

(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 Pohlig-Hellman keys is simply a matter of concatenating a 1 with a random number of suitable size in order to ensure they are odd (and thus a unique inverse exists). q should not be used as a key, but the probability of that happening is so low that we ignore it.

The encryption key e is stored in the simple universal 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 the resultant "overall" key must be odd, so if the result of decrypting the universal block is

even we add q to obtain the correct 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 $e \in_u$ Zp, e mod 2 = 1; and random k_0, $k_u \in$ Zq.

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 $e' \in$ Zp, e' mod 2 = 1; and random k_0', $k_u' \in$ Zq.

Output is a ciphertext 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 $\alpha_u / \ \beta_u^x \bmod q = 1$ 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 $\alpha_u / \ \beta_u^x \bmod q \neq 1$, 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, and permits unlimited re-encryptions.

4 Futher Improvements

4.1 Implementation notes

The first 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 values are 1,024 bits.

The second security parameter is the length of the Pohlig-Hellman keys used. A keylength long enough to make low Hamming count attacks approximately as hard as finding discreet logarithms over the whole k bits is sufficient. For the 1024 bit example a keysize of 160 bits is about right [4]. This shorter key speeds up encryption and re-encryption considerably.

There is no useful subgroup of prime order in the Pohlig-Hellman field, which means that decryption will involve exponentiation to a full k-bit exponent, making decryption slower than it otherwise might be. However we assume that de-

cryption 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 probably true in most situations where re-encryption is useful.

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. A subgroup of order around 160 bits will give a better than order of magnitude performance improvement here [ref] without impact on security.

There may be some chosen plaintext and adaptive chosen plaintext attacks, depending on precise inplementation - one simple solution is to pre-encrypt the plaintext with a symmetric cipher and a random IV, using either a shared or a secret key.

The Pohlig-Hellman cipher is however much still much slower than a modern symmetric cipher. Roughly speaking, a modern 3 GHz machine will only be able to re-encrypt at around 50 kB/s even with highly optimised code, ie it will just about handle re-encryption over a 1 Mb/s half-duplex broadband connection, or a dual 3 Ghz will re-encrypt a T1 line at an average load. Anything more demanding will need better or specialised hardware.

4.2 Future directions

While it would be computationally advantageous to replace Pohlig-Hellman with a suitable symmetric cipher, no 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 [5] based on the group property.

Such a cipher would potentially have other uses, in Atomic Proxy Cryptography [6], trusted parties [7], 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 than a simple system. The removal of all limitations on the number of possible re-encryptions makes it far more widely useful than the previous hybrid system.

While not as fast as a symmetric cipher hybrid system, it is suitable for use in situations where small data size is paramount and computing cycles are plentiful. Where modern computers are used to pass data over internet connections, this will often be the case.

Acknowledgements: Discussions with Debra Cook, Paul Rubin and John Malley have made me aware of other research, and the accepted names for things I've reinvented. I thank them for their time and kindness.

References:

[1]Universal Re-encryption for Mixnets.
by P. Golle, M. Jakobsson, A. Juels and P. Syverson.
To be presented at RSA 2004, Cryptographer's Track
http://citeseer.nj.nec.com/golle02universal.html

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

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

[4] (Wei Dai/NIST)

[5] Is the Data Encryption Standard a Group?
by Burton S. Kaliski Jr,. Ronald L. Rivest, and Alan T. Sherman Eurocrypt 85

[6] by Matt Blaze

[7] (nyu paper)