Conversion and Proxy Functions for Symmetric Key Ciphers

Debra L. Cook Angelos D. Keromytis
Department of Computer Science, Columbia University
{*dcook,angelos*}*@cs.columbia.edu*

Abstract

As a general design criterion, a symmetric key cipher should not be closed under functional composition due to the implications on the security of the cipher. However, there are scenarios in which this property is desirable and can be obtained without reducing the security of a cipher by increasing the computational workload of the cipher. We expand the idea of a symmetric key cipher being closed under functional composition to a more general scenario where there exists a function that converts the ciphertext resulting from encryption under a specific key to the ciphertext corresponding to encryption with another key. We show how to perform such a conversion without exposing the plaintext. We discuss the tradeoff between the computational workload and security, and the relationship between such conversions and proxy cryptography. We conclude with a description of some practical applications of our results.

Keywords: Symmetric Key Cipher Design, Functional Composition, Proxy Cryptography

1 Introduction

We expand the idea of a symmetric key cipher being closed under functional composition to a more general scenario in which there exists a function that converts the ciphertext resulting from encrypting with a specific key to the ciphertext corresponding to encrypting with another key. As a general design criterion, a symmetric key cipher should not be closed under functional composition due to the implications on the security of the cipher. However, there are scenarios in which this property is desirable and can be obtained without reducing the security of a cipher by increasing the computational workload of the cipher. We discuss the tradeoff between the computational workload and security. We show how to construct from any symmetric key cipher a cipher that allows an entity to convert ciphertexts between two keys without exposing the plaintext and describe practical applications of the results. We also discuss the relationship between such conversions and proxy cryptography.

The motivation for our work arises from a conversion problem. Is it possible to define a symmetric key cipher that allows for converting the encryption of text, $E_{k1}(P)$, under one key to the encryption of the text under another key, $E_{k2(}P)$, with fewer computations than what is required for decrypting with key $k1$ then encrypting with key $k2$? Consider the case of a virtual private network (VPN) gateway transmitting data between users A and B. The gateway shares $k1$ with A and $k2$ with B. A and B do not share any key material. With existing symmetric key ciphers, the gateway must perform the conversion by decrypting with $k1$ then encrypting with $k2$. Specifically, A computes $C1 = E_{k1}(P)$ and sends $C1$ to the gateway. The gateway computes $C2 = E_{k2}(D_{k1}(C1))$ and sends $C2$ to B who computes $P = D_{k2}(C2)$. Is there a conversion function G taking a key kg such that

(I) $\quad G_{kg}(E_{k1}(P)) = E_{k2}(P) \quad \forall P$

where kg depends on $k1$ and $k2$, and G requires less work than applying both E and D with some acceptable tradeoffs? In this application, the goal is to decrease the conversion time. The gateway may have sufficient information to obtain P and may or may not expose P during the conversion; in some situations it is desirable for part of P to be obtainable for inspection. The existence of a function G as shown in (I) has significant implications on the security of the cipher, which we will discuss.

We are also interested in conversions that prohibit the gateway from obtaining the ciphertext in situations where there is no need for it to have access to the plaintext. While proxy cryptography is oriented towards ensuring the intermediate entity (the proxy) performing the conversion cannot obtain the plaintext, the work that exists in this area is focused on public key ciphers; whereas, we require a symmetric key cipher in order for the conversion to be applicable in situations involving larger quantities of data and faster processing than what can be supported with public key ciphers. Okamoto and Mambo [8] introduced the notion of proxy cryptography. This was further explored by Blaze, *et al.* in [1]. Proxy cryptography allows two parties to publish a key that the proxy will use to convert ciphertext received from one party into ciphertext that can be decrypted with the other party's private key without the proxy being able to decrypt the text. Okamoto and Mambo provide a means of conversion for El Gamal [2] and RSA [9] that is more efficient than decrypting and re-encrypting. A

related problem dealing with "re-encryption" is discussed in [3]. In [6], Ivan and Dodis restated the definition from [1]. While [6] includes a generic form of proxy cryptography that works with symmetric key systems, we point out that the construction is merely the encryption mechanism used in onion routing [5] with symmetric keys. For entity A to send data to entity B through a proxy, A performs double encryption using two different keys on the plaintext. A shares the key used for the first encryption with B and the key used for the second encryption with the proxy entity. Upon receiving the data from A, the proxy performs one application of decryption and forwards the data to B, which performs one decryption to retrieve the plaintext. This construction does not solve our original problem of reducing the workload, it only moves the work from the gateway to the originating entity A. Furthermore, it creates an undesirable situation in which A and B are required to share key material. If A and B can securely share a secret key, it is not necessary for them to communicate via a proxy except in cases where they are trying to hide the fact that they are communicating and for which solutions exist.

The contributions of our work are the following. First, we extend the results in [7] concerning the security of a symmetric key cipher that is closed under functional composition to the more general scenario of a symmetric key cipher for which there exists a conversion function. Second, we introduce the term *secure conversion crypto-system* to refer to a symmetric key cipher combined with a conversion function which allows converting between encryptions under different keys without exposing the plaintext. We construct secure conversion crypto-systems from any existing symmetric key cipher. Third, we prove our conversion crypto-system constructions are optimal in terms of the order of computational work compared to the security of the underlying cipher utilized in the construction, and discuss tradeoffs between the workload and security in terms of the underlying cipher. Finally, we discuss possible applications of conversion and secure conversion crypto-systems.

Paper Organization: In Section 2 we define our notation and introduce the terms for conversion crypto-systems. In Section 3 we review the attacks from [7] on symmetric key ciphers which are closed under functional composition. In Section 4 we generalize the attacks from [7] to symmetric key ciphers for which conversion functions exist. In Sections 5 and 6 we present constructions and applications of conversion crypto-systems. Section 7 concludes the paper.

2 Notation

We use the following notation. i indicates any alphanumeric symbol.

- P denotes plaintext.
- C, Ci denotes ciphertext.
- S denotes a symmetric key cipher.
- K denotes the key space for a cipher.
- $|K|$ denotes the size of the key space.
- k, ki, k_i denote keys.
- E, D denote encryption and decryption, respectively.
- $E_k(P), D_k(C)$ denote encryption and decryption, respectively, using key k.
- $G_{kg}(X)$ denotes a function G that takes key kg and input X.
- KS, KS_i denotes a key stream

We introduce the terms (secure) conversion function and (secure) conversion crypto-system. Unless otherwise stated, our use of the following terms is limited to symmetric key ciphers within the context of this paper.

- Conversion function refers to a function $G_{kg}(X)$ that converts $E_{k1}(P)$ to $E_{k2}(P)$ $\forall P$ for any keys $k1, k2$. Specifically, $G_{kg}(E_{k1}(P)) = E_{k2}(P)$ $\forall P$ where kg is dependent on $k1$ and $k2$. There is no restriction on whether the conversion function exposes P during the conversion and on whether the entity performing the conversion has sufficient information to obtain P. Without loss of generality, kg is a single parameter. If G requires multiple parameters that are dependent on $k1$ and/or $k2$, all parameters are concatenated to form kg.
- Conversion crypto-system refers to a cipher for which a conversion function exists along with the conversion function.
- Secure conversion function refers to a conversion function that does not expose P during the conversion. There is no restriction on whether or not the entity performing the conversion has sufficient information to obtain P.
- Secure conversion crypto-system refers to a cipher for which a secure conversion function exists along with the secure conversion function.
- Proxy function refers to a function $G_{kg}(X)$ that converts $E_{k1}(P)$ to $E_{k2}(P)$ $\forall P$ for any keys $k1, k2$ without exposing P during the conversion and which does not provide the entity performing the conversion sufficient information to obtain P. This is a special case of secure conversion functions.
- Proxy crypto-system refers to a cipher for which a proxy function exists combined with the proxy function. Proxy crypto-systems are a subset of secure conversion crypto-systems.
- Conversion entity and converter will be used interchangeably to refer to the entity performing G.
- Proxy refers to the entity performing G in a proxy crypto-system. This is a special class of conversion entities.

- Effective key length: K_{eff} is a parameter defined in terms of the key length, $|k|$, which indicates the amount of work required to successfully attack the cipher compared to that of an exhaustive search of all keys. A cipher for which an exhaustive search of the keys is the best known attack requires $\bigcirc(2^{|k|})$ work and has an effective key length of $|k|$.

3 Closure Under Function Composition

Before describing conversion systems, we review why closure under functional composition is an undesirable property for symmetric key ciphers. Within the context of determining whether not DES [4] is a group, Kaliski, *et al.* proved in [7] that any symmetric key cipher that is closed under functional composition is vulnerable to a known plaintext attack requiring $\bigcirc(2^{|k|/2})$ work as opposed to $\bigcirc(2^{|k|})$ work required for an exhaustive key search. We will write $\bigcirc(2^{|k|/2})$ as $\bigcirc(|K|^{1/2})$, where K is the keyspace. Two methods of known plaintext attacks were described in [7], with a tradeoff between the memory and the time required to decrypt additional ciphertexts. We provide a brief summary of these attacks. They do not provide the actual secret key, but instead provide a series of keys which, when applied in order, produce the same results as the secret key.

Assume a symmetric key cipher, S, with keyspace, K, is closed under functional composition. E and D are the encryption and decryption functions for S, respectively. Then $\forall\ k1, k2 \in K, \exists\ k3 \in K$ such that:

(II) $\quad E_{k3}(E_{k1}(P)) = E_{k2}(P) \quad \forall\ P$

The first attack produces a pair of keys, $(k1, k3)$, which can be used in place of $k2$ as indicated by (II). Choose two sets of r keys $KA = \{k_{a1}, k_{a2}, ...k_{ar}\}$ and $KB = \{k_{b1}, k_{b2}, ...k_{br}\}$ from K. For all pairs (k_{ai}, k_{bi}), determine if (II) holds via a meet in the middle attack. Let $C = E_{k2}(P)$. Compute $E_{k_{ai}}(P)\ \forall\ k_{ai}$ and $D_{k_{bj}}(C)\ \forall\ k_{bj}$ and search for matches. Set $k1$ to the k_{ai} and $k3$ to the k_{bj} that produce the match. Test with additional plaintexts to ensure the match does not hold for only a specific P. This attack will produce a pair of keys that are equivalent to the single key $k2$ as opposed to finding $k2$. The key pair can be used to decrypt additional ciphertexts that have been encrypted with $k2$. Obviously if either $E_{k_{ai}}(P) = C$ or $D_{k_{bj}}(C) = P$ is found during the search, then $k2$ has been found.

The result derives from a meet-in-the-middle variation of the Birthday Paradox using two samples X and Y. If X and Y are of size r, and are drawn at random from $|K|$ elements with each element drawn independently with probability $\frac{1}{|K|}$, then there are $\binom{|K|}{r}$ ways to select X and $\binom{|K|-r}{r}$ ways to select Y such that $X \bigcap Y = \emptyset$ and $\binom{|K|}{r}^2$ ways to select X and Y. Thus, the chance that X and Y do not intersect is:

(III) $Pr(X \bigcap Y = \emptyset) = \frac{[(|K|)(|K|-1)...(|K|-2r+1)]}{[((|K|)(|K|-1)...(|K|-r+1))^2]}$

If $r = \alpha(|K|^{1/2})$ for some constant $\alpha > 0$, then $Pr(X \bigcap Y = \emptyset) \approx e^{-3\alpha^2}$ for sufficiently large $|K|$. The cipher S is transformed into this variation of the Birthday Paradox by using KA and KB as the two samples and defining intersection to mean there is a $k1 \in KA$ and a $k3 \in KB$ such that $E_{k3}(E_{k1}(P)) = E_{k2}(P) \forall P$. The probability of finding a $(k1, k3)$ is approximately $1 - e^{-3\alpha^2}$ and can be made as close to 1 as desired by increasing α. The attack requires $\bigcirc(|K|^{1/2})$ time and memory.

The second attack in [7] is referred to as a cycling attack. While it requires less memory than the first attack, it produces a series of keys that require $\bigcirc(|K|^{1/2})$ time for decryption in contrast to the two keys produced by the first method. The idea is to obtain some series of encryptions and decryptions that started with P and C, respectively, and intersect. Specifically, $E_{k_{ai}}(E_{k_{ai-1}}...(E_{k_{a2}}(E_{k_{a1}}(P))))$ $= D_{k_{bj}}(D_{k_{bj-1}}...(D_{k_{b2}}(D_{k_{b1}}(C))))$ for randomly chosen $k_{ai}, k_{bj} \in K$. The result can be verified by testing with a few additional plaintexts. An average of $|K|^{1/2}$ steps are needed before the two sides match. Like the first attack, $k2$ is not found. However, the equivalent key for $k2$ that is produced this time is the series of k_{ai}'s and k_{bj}'s. To decrypt additional ciphertexts encrypted with $k2$, $D_{k_{a1}}(D_{k_{a2}}...(D_{k_{ai-1}}(D_{k_{ai}}(D_{k_{bj}}(D_{k_{bj-1}}$ ... $(D_{k_{b2}}(D_{k_{b1}}(C)))))))$ must be computed. $\bigcirc(|K|^{1/2})$ space is needed for storing the keys. Overall, the algorithm requires $\bigcirc(|K|^{(1+w)/2})$ time and space for small w.

4 Generalization of Attacks to Conversion Functions

We now generalize the first attack from [7] to an attack on a symmetric key cipher for which a conversion function exists. A symmetric key cipher which is closed under functional composition (which [7] addressed) is a special case of a symmetric key cipher for which a conversion function exists, specifically the case where the conversion function is the cipher. Notice that if a conversion function exists, then $|KG| \geq |K|$. If not, then the existence of a kg for every pair of keys $k1, k2 \in K$ requires at least one kg to map a $k1$ to more than one $k2$.

Lemma I: For a symmetric key cipher S with keyspace K and encryption function E, if there exists a function G taking parameter $kg \in KG, |KG| = |K|$, and $\forall k1, k2 \in K$, $\exists$ a kg for which $G_{kg}(E_{k1}(P)) = E_{k2}(P)\ \forall P$ then there exists a $\bigcirc(|K|^{1/2})$ known plaintext attack on S.

Proof: The cipher S is transformed as before into the variation of the Birthday Paradox by using KA and KB as the two samples and defining intersection to mean there is a k_{ai} in KA and a k_{bj} in KB such that $G_{k_{bj}}(E_{k_{ai}}(P)) =$

$E_{k2}(P)$. Now KB is from KG instead of K. Since $|KG| = |K|$, (III) holds with $X = KA$ and $Y = KB$.

Lemma II: For a symmetric key cipher S with keyspace K and encryption function E, if there exists a function G taking parameter $kg \in G$, $|KG| \geq |K|$, $\forall k1, k2 \in K$ $\exists$ one and only one kg for which $G_{kg}(E_{k1}(P)) = E_{k2}(P)$ $\forall P$, and the work of G is $\bigcirc$(work of E), then there exists a known plaintext attack on S that is $\bigcirc(|K|^{1/2})$.

Proof: Let KA denote the set of r keys chosen from K and let KB denote the set of r keys chosen from KG. When $|KG| \geq |K|$, for any element in KB, there may or may not be an element in K that we can combine with it to obtain a key equivalent to $k2$. Without loss of generality, choose KA first. There are r elements in KG that can create a match with some element of K and $\binom{|KG|-r}{r}$ ways to select r elements from KG such that no match is formed. Let $Pr[k1, k3]$ be the probability of finding a $k1$ from KA and a $k3$ from KB, and let $Pr[\overline{(k1, k3)}]$ denote the probability of not finding such a pair.

$Pr[k1, k3] = 1 - Pr[\overline{(k_1, k_3)}]$

$Pr[\overline{(k_1, k_3)}] =$

$\binom{|K|}{r}\binom{|KG|-r}{r} / \binom{|KG|}{r}\binom{|K|}{r} \geq \binom{|K|}{r}\binom{|K|-r}{r} / \binom{|K|}{r}^2$

with equality holding when $|KG| = |K|$, and

$Pr[(k1, k3)|\ |KG| \geq |K|] =$

$1 - Pr[\overline{(k_1, k_3)}|\ |KG| \geq |K|]$

$\leq 1 - Pr[\overline{(k_1, k_3)}|\ |KG| = |K|]$

As the probability of success decreases, the number of keys to try on average before finding a match increases from the $(|K|^{1/2})$ average obtained when $|KG| = |K|$.

Lemma I implies that for any symmetric key cipher with a conversion function taking parameters (keys) of the same length as the key length, $|k|$, of the cipher, the effective key length of the cipher is less than or equal to $\frac{1}{2}|k|$. Equality may not hold due to the possibility that G may require less work than E or provide some other speedup aside from the square root reduction in the number of keys to try to the extent that a brute force attack is $\bigcirc(2^{\frac{1}{2}|k|-w})$ as opposed to $\bigcirc(2^{\frac{1}{2}|k|})$ for some $w > 0$. To obtain security comparable to a brute force attack over all keys of length k, $\bigcirc(2^{|k|})$, the key length must at least be doubled to $2|k|$.

5 Conversion Crypto-systems

5.1 Secure Conversion Construction

We now define a general construction for symmetric key secure conversion crypto-systems. First, we consider all symmetric key ciphers and provide two variations for the defining keys. The variations differ in the workload required of each entity and in which entities share key material. Second, we provide a variation that is restricted to stream ciphers and offers advantages over the general construction. A conversion function can be constructed (trivially) for any symmetric key cipher by defining $G_{kg}(C)$ to be $E_{k_2}(D_{k_1}(C))$ to convert $E_{k_1}(P)$ to $E_{k_2}(X)$. This is not a secure system under our definition due to the fact that the plaintext is exposed during the conversion.

We define the following key format and function for use in our constructions:

- Let $k = (k_1, k_2, flag_1, flag_2)$ for keys k_1, k_2 with $|k_1| = |k_2|$ and single bit flags $flag_1, flag_2$. The $flag_i$ and k_i values will be used to denote whether to encrypt, decrypt or do nothing. A $flag_i$ value of 0 indicates to encrypt with key k_i and a value of 1 indicates to decrypt with key k_i. If k_i is null, do nothing.
- Let $F_k(X)$ denote $F(E, D, X, k)$ and be defined as applying E or D, as indicated by k, to X. From k, k_1 and $flag_1$ are used for the first application, and k_2 and $flag_2$ are used for the second application of E or D.

The first method for defining the keys requires each pair of the three entities to share some key material. For any symmetric key cipher, $F_k(X)$ is a symmetric key secure conversion crypto-system with keys $k_A = (k_{ab}, k_a, 0, 0)$, $k_g = (k_a, k_b, 1, 0)$ and $k_B = (k_b, k_{ab}, 1, 1)$. To send plaintext P from A to B via a converter, A computes $C1 = F_{kA}(P)$, which is the equivalent of the double encryption $E_{k_a}(E_{k_{ab}}(P))$, and sends $C1$ to the converter. The converter computes $C2 = F_{kg}(C1)$, which is the equivalent of $E_{k_b}(D_{k_a}(C1))$, and sends $C2$ to B. To obtain P, B computes $F_{kB}(C2)$, which is the equivalent of the double decryption $D_{k_{ab}}(D_{k_b}(C2))$.

The second method for defining the keys is basically onion routing. It does not require any shared key material between the converter and B in order for B to receive messages from A. The key values are $k_A = (k_{ab}, k_a, 0, 0)$, $k_g = (k_a, null, 1, 0)$ and $k_B = (k_{ab}, null, 1, 0)$.

There are subtle differences between the two sets of keys. Even though the first construction imposes twice the workload on each entity compared to a single application of a cipher, whereas the second construction only increases the workload of one entity, the first construction offers a potential advantage in how the key material is shared. While in both cases A and B must share some key material, in the first case no entity has the entire key of any other entity. In the second case, A has the entire key used by each of the other two entities. B must share some key material with every entity with which it exchanges data. In the first case, B and the converter share k_b for use in all communications and B can use k_{ab} with some $A' \neq A$ if desired. In the extreme case, k_{ab} can be public, though this has a downside in that if it is possible for an adversary to access memory within the converter as $F_{kg}(C1)$ is being computed and obtain $D_{k_a}(C1)$, then the adversary can compute $D_{k_{ab}}(D_{k_a}(C1))$ and obtain P. In the second case B must establish a shared secret key with every entity with which

it wishes to communicate (eliminating the need for a conversion entity). B cannot re-use k_{ab} with some $A' \neq A$ because then A can decrypt every message A' sends to B. While the construction does not require shared key material between B and the converter for B to receive messages, they must share key material if B is also allowed to send messages.

An alternative way of viewing the first construction requires a symmetric key cipher that has r rounds. In the first method, let $E^{r1,r2}_{k1,k2}$ denote running $r1$ rounds of encryption using key $k1$ followed by $r2$ rounds of encryption using key $k2$. Let F and the keys be defined as before. Each entity will compute two key expansions, one for each part of its key, use the first key expansion for a specified number of rounds and use the second key expansion for the remaining number of rounds. For example, A will run $r1$ rounds of encryption with the expanded k_{ab} and $r2 = r - r1$ rounds of encryption with the expanded k_a. The converter will run $r2$ rounds of decryption using the expanded k_a and $r2$ rounds of encryption using the expanded k_b. B will decrypt by running $r1$ rounds with the expanded k_{ab} and $r2$ rounds with the expanded k_b. The converter requires a total of $2 * (r2)$ rounds. Setting $r1 = r2$ results in each entity running r rounds. Specifically,

(IV) A computes $C1 = E^{r1,r2}_{k_{ab},k_a}(P)$, the converter computes $C1 = E^{r2}_{k_b}(D^{r2}_{k_a}(C1))$, and B computes $P = D^{r2,r1}_{k_b,k_{ab}}(C2)$.

In this case, instead of viewing the resulting system as having twice the work of the underlying cipher but the security equivalent to that of the underlying cipher, the system can be thought of as having the same amount of work as the underlying cipher with an effective key length of $\frac{1}{2}$ the key length of the underlying cipher. This construction is equivalent to using two applications of a reduced round version of the underlying cipher, thus the number of rounds chosen for each step must be large enough avoid making the reduced round version susceptible to other attacks, such as differential and linear cryptanalysis.

If the cipher is a stream cipher (or a block cipher run in a stream cipher mode, for example, OFB or CTR) such that $C = KS \oplus P$, where KS is the key stream, then it is not necessary for A and B to share any key material if the keys are $k_A = (k_a, null, 0, 0)$, $k_g = (k_b, k_a, 1, 1)$ and $k_B = (k_b, null, 1, 0)$. Let KS_a and KS_b denote the key streams produced by E when using k_a and k_b respectively. A computes $C1 = F_{k_A}(P)$, which is the equivalent of $KS_a \oplus P$. The converter computes $C2 = F_{kg}(C1)$, which is the equivalent of $KS_a \oplus KS_b \oplus C1$. B computes $F_{k_A}(P)$, which is the equivalent to $KS_b \oplus C2$. The converter can compute $KS_a \oplus KS_b \oplus C1$ either by computing $KS_b \oplus C1$ or $KS_a \oplus KS_b$ first. There are several advantages to this construction. First, A and B do not share any key material. Second, P is not revealed during the conversion, though the converter does have the information needed to obtain P. Third, it is not necessary that A and B incur the minor overhead related to utilizing F in place of the underlying stream cipher. F may be run only by the converter. A and B can just apply the stream cipher directly. We discuss the potential use of the stream cipher version in Section 6

Since secure conversion functions are special cases of conversion functions, by Lemma 1, if the secure conversion function takes key material kg of length $|k|$ for the symmetric key cipher (or is the same function as the cipher), then $|KG| = |K|$ and an attack of $\bigcirc(|K|^{1/2})$ exists and $K_{eff} \leq \frac{1}{2}|k|$. In our constructions, the key length for both the encryption and conversion functions are twice that of the underlying cipher's key length (excluding the flag bits in KG), resulting in $K_{eff} = |k|$ and thus are the best possible in terms of security versus key length.

5.2 Symmetric Key Proxy Crypto-Systems

Our general construction of a symmetric key crypto-system satisfies the definition of a proxy crypto-system. However, since it requires A and B to share key material, it is not obvious what benefit this system provides outside of scenarios aimed at preventing traffic analysis. The bounds we provide for conversion crypto-systems apply to symmetric key proxy crypto-systems because the proxy case is a subset of the general conversion case.

The stream cipher specific version does not satisfy the definition of a proxy crypto-system because the converter has enough information to obtain the plaintext even though it does not do so by definition of the crypto-system. The converter is a proxy if it can be told kg without knowing k_A and/or k_B. This may be possible if the crypto-system is defined in a manner in which kg is not merely the concatenation of key material from the endpoints. Through the use of an external device, such as a smart card or a secure crypto processor, which given k_A and k_B, (or already contains k_a and k_b and is provided some identifying information indicating to use the keys for A and B), returns k_g, it may be possible to provide the appropriate key to the converter without revealing key material from A and B.

6 Applications

In regards to our original motivation of reducing the converter's workload, the generic symmetric key secure conversion crypto-system construction solves this when the system is written in the form defined in (IV) and the number of rounds required of the converter is below that of the original cipher (at a cost of reduced security). Furthermore, even in its general form (as opposed to requiring a cipher consisting of rounds), it offers the advantage that the plaintext is not exposed during the conversion. In existing applications that convert ciphertext via the basic mode of decrypting then encrypting, whether or not the plaintext is accessible temporarily during the conversion (in part due to intermediate

results being written to memory), depends on the application and implementation. With a secure conversion crypto-system, it is not a concern if intermediate results are written to temporary files or insecure memory during the conversion because it is still encrypted under one key.

The main disadvantage of the construction is that A and B must share key material, which results in implementation issues. If the converter needs to inspect some of the packets, it must establish the key material shared between A and B. Otherwise, if A and B can establish shared key material on their own, one of them must send the shared key to the converter. However, if A and B can establish a shared key, it can be argued there is no need for a conversion entity unless it is used in preventing traffic analysis. If the converter does not need to inspect any packets, but establishes the shared key material between A and B, then the converter can obtain the plaintext even though it does not do so if the algorithm is applied as specified. This is not preferred in any application that requires a proxy as opposed to just a secure converter. Scenarios where a converter may need to inspect packets include VPN gateways and firewalls. In some applications, the gateway or firewall replaces parts of the contents, such as in application aware NAT where IP addresses embedded in the application data are replaced. A scenario where the converter does not need to inspect packets is a file system in which the files are encrypted under one key and sent to a requesting user encrypted with the user's key.

The construction specific to stream ciphers does not require A and B to share key material. It also provides the advantage of being a secure conversion system, with the converter being able to inspect data if required but otherwise not exposing the plaintext during the conversion. There is no overhead for A and B, as they both continue to perform a single application of the stream cipher. There are mechanisms by which the converter cannot obtain the plaintext. For example, a hardware implementation that runs two instances of the key stream generator and XORs their outputs then makes the resulting key stream available for XORing with the data. The keys may be configurable in hardware and not accessible by any software applications on the converter.

7 Conclusions

We have shown that for any symmetric key cipher S with key space K, if there exists a conversion function requiring a key whose length is the same as the length of the elements of K and whose work is $\bigcirc(S)$ then there exists an $\bigcirc(|K|^{1/2})$ attack. If the key length for the conversion function is greater than the length of the keys in K and/or the work of the conversion function is $\Omega(S^c)$ for some $c > 1$, then the existence of an attack requiring less work than an exhaustive key search on K depends on the size of the conversion function's key space and its workload. Our work poses the question of whether or not an actual symmetric key cipher can be designed with a conversion function (other than decrypt and encrypt) such that a variable tradeoff between workload and security (effective key length) can be set per application.

We provided methods for constructing a secure conversion crypto-system from any symmetric key cipher that can convert text between the ciphertext corresponding to encryption under one key to that corresponding to encryption under a second key without exposing the plaintext during the conversion. The work of the resulting system is twice that of the underlying cipher with an effective key length of that of the underlying cipher. If the underlying cipher consists of multiple rounds, the conversion crypto-system can be defined such that the number of rounds performed by the sender, converter and receiver vary according to the desired level of security and workload. If the underlying cipher is a stream cipher, the conversion can be implemented in a manner requiring no additional work on the sending and receiving entities, and either allowing or disallowing the converter access to the plaintext. The security is unchanged from that of the basic stream cipher with the added benefit that the plaintext is not exposed during the conversion.

References

[1] M. Blaze, G. Bleumer, and M. Strauss. Atomic Proxy Cryptography and Protocol Divertibility. In *Proceedings of EUROCRYPT '98, LNCS 1403, Springer-Verlag*, May 1998.

[2] T. ElGamal. A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985.

[3] P. Fairbrother. An Improved Construction for Universal Reencryption. In *Workshop in Privacy Enhancing Technologies*, May 2004.

[4] FIPS 46-3. Data Encryption Standard (DES), 1999.

[5] D. Goldschlag, M. Reed, and P. Syverson. Onion Routing for Anonymous and Private Internet Connections. *Communications of the ACM (USA)*, 42(2):39–41, 1999.

[6] A. Ivan and Y. Dodis. Proxy Cryptography Revisted. In *Proceedings of the Network and Distributed Systems Security Symposium (NDSS)*, February 2003.

[7] B. Kaliski, R. Rivest, and A. Sherman. Is the Data Encryption Standard a Group? *Journal of Cryptology*, pages 3–36, 1988.

[8] Okamoto and Mambo. Proxy Cryptosystems: Delegation of Power to Decrypt Ciphertexts. *IEICE Trans. Fund. Eletronic Communications and Comp Sci. E80-A/1*, pages 54–63, 1997.

[9] RSA Laboratories. *PKCS #1: RSA Encryption Standard*, version 1.5 edition, November 1993.