Increasing the Lifetime of a Key: A Comparative Analysis of the Security of Re-keying Techniques

Michel Abdalla and Mihir Bellare

Dept. of Computer Science & Engineering, University of California at San Diego,
9500 Gilman Drive, La Jolla, California 92093, USA.
{mabdalla, mihir}@cs.ucsd.edu
www-cse.ucsd.edu/users/{mabdalla, mihir}

Abstract. Rather than use a shared key directly to cryptographically process (e.g. encrypt or authenticate) data one can use it as a master key to derive subkeys, and use the subkeys for the actual cryptographic processing. This popular paradigm is called re-keying, and the expectation is that it is good for security. In this paper we provide concrete security analyses of various re-keying mechanisms and their usage. We show that re-keying does indeed "increase" security, effectively extending the lifetime of the master key and bringing significant, provable security gains in practical situations. We quantify the security provided by different re-keying processes as a function of the security of the primitives they use, thereby enabling a user to choose between different re-keying processes given the constraints of some application.

1 Introduction

Re-keying (also called key-derivation) is a commonly employed paradigm in computer security systems, about whose security benefits users appear to have various expectations. Yet the security of these methods has not been systematically investigated. Let us begin with some examples that illustrate usage, commonly employed implementations, and motivation for re-keying, and see what security issues are raised. We then go on to our results.

Re-keyed encryption. Say two parties share a key K, and want to encrypt data they send to each other. They will use some block cipher based mode of operation, say CBC. The straightforward approach is to use K directly to encrypt the data. An often employed alternative is re-keyed encryption. The key K is not used to encrypt data but rather viewed as a master key. Subkeys $K_1, K_2, K_3, \ldots$ are derived from K, by some process called the re-keying process. A certain number l of messages are encrypted using K_1 and then the parties switch to K_2. Once l messages have been encrypted under K_2 they switch to K_3 and so on.

Examples of re-keying methods. Many different re-keying methods are possible. Let us outline two of them. In each case $F(\cdot,\cdot)$ is a map that takes a k-bit key κ and k-bit input x to a k-bit output $F(\kappa, x)$. (This might be implemented via a block cipher or a keyed hash function.) The *parallel* method consists of

T. Okamoto (Ed.): ASIACRYPT 2000, LNCS 1976, pp. 546–559, 2000.

setting $K_i = F(K, i)$ for $i = 1, 2, \ldots$. The *serial* method sets $k_0 = K$ and then sets $K_i = F(k_{i-1}, 0)$ and $k_i = F(k_{i-1}, 1)$ for $i = 1, 2, \ldots$. Many other methods are possible, including hybrids of these two such as tree-based re-keying [1].

Why re-key? Common attacks base their success on the ability to get lots of encryptions under a single key. For example differential or linear cryptanalysis [10,17] will recover a DES key once a certain threshold number of encryptions have been performed using it. Furthermore, most modes of operation are subject to birthday attacks [3], leading to compromise of the privacy of a scheme based on a block cipher with block size k once $2^{k/2}$ encryptions are performed under the same key. Typically, the birthday threshold is lower than that of the cryptanalytic attacks.

Thus, if encryption is performed under a single key, there is a certain maximum threshold number of messages that can be safely encrypted. Re-keying protects against attacks such as the above by changing the key before the threshold number of encryptions permitting the attack is reached. It thus effectively extends the lifetime of the (master) key, increasing the threshold number of encryptions that can be performed without requiring a new exchange of keys.

Questions. Although re-keying is common practice, its security has not been systematically investigated. We are interested in the following kinds of questions. Does re-keying really work, in the sense that there is some *provable* increase in security of an application like re-keyed encryption described above? That is, can one prove that the encryption threshold —number of messages of some fixed length that can be safely encrypted— increases with re-keying? How do different re-keying processes compare in terms of security benefits? Do some offer more security than others? How frequently should the key be changed, meaning how should one choose the parameter l given the parameters of a cryptographic system?

High level answers. At the highest level, our answer to the most basic question (does re-keying increase security?) is "YES." We are able to justify the prevailing intuition with concrete security analyses in the provable security framework and show that re-keying, properly done, brings significant security gains in practical situations, including an increase in the encryption threshold. Seen from closer up, our results give more precise and usable information. We quantify the security provided by different re-keying processes as a function of the security of the primitives they use. This enables comparison between these processes. Thus, say a user wants to encrypt a certain amount of data with a block cipher of a certain strength: our results can enable this user to figure out which re-keying scheme to use, with what parameters, and what security expectations.

Re-keyed CBC encryption. As a sample of our results we discuss CBC encryption. Suppose we CBC encrypt with a block cipher F having key-length and block-length k. Let's define the encryption threshold as the number Q of k-bit messages that can be safely encrypted. We know from [3] that this value is $Q \approx 2^{k/2}$ for the single-key scheme. We now consider re-keyed CBC encryption under the parallel or serial re-keying methods discussed above where we use the same block cipher F as the re-keying function. We show that by re-keying every $2^{k/3}$ encryptions —i.e. set the subkey lifetime $l = 2^{k/3}$— the encryption threshold increases to $Q \approx 2^{2k/3}$. That is, one can safely encrypt significantly more data by using re-keying. The analysis can be found in Section 3.

Overview of approach and results. Re-keying can be used in conjunction with any shared-key based cryptographic data processing. This might be data encryption, under any of the common modes of operation; it might be data authentication using some MAC; it might be something else. We wish to provide tools that enable the analysis of any of these situations. So rather than analyze each re-keyed application independently, we take a modular approach. We isolate the re-keying process, which is responsible for producing subkeys based on a master key, from the application which uses the subkeys. We then seek a general security attribute of the re-keying process which, if present, would enable one to analyze the security of any re-keying based application. We suggest that this attribute is pseudorandomness. We view the re-keying process as a stateful pseudorandom bit generator and adopt a standard notion of security for pseudorandom bit generators [11,18]. We measure pseudorandomness quantitatively, associating to any re-keying process (stateful generator) $\mathcal{G}$ an advantage function $\mathsf{Adv}^{\mathrm{prg}}_{\mathcal{G},n}(t)$, which is the maximum probability of being able to distinguish n output blocks of the generator from a random string of the same length when the distinguishing adversary has running time at most t. We then analyze the parallel and serial generators, upper bounding their advantage functions in terms of an advantage function associated to the underlying primitive F. See Section 2.

To illustrate an application, we then consider re-keyed symmetric encryption. We associate a re-keyed encryption scheme to any base symmetric encryption scheme (e.g. CBC) and any generator. We show how the advantage function of the re-keyed encryption scheme can be bounded in terms of the advantage function of the base scheme and the advantage function of the generator. (The advantage function of an encryption scheme, whether the base or re-keyed one, measures the breaking probability as a function of adversary resources under the notion of left-or-right security of [3].) Coupling our results about the parallel and serial generators with known analyses of CBC encryption [3] enables us to derive conclusions about the encryption threshold for CBC as discussed above. See Section 3.

Security of the parallel and serial generators. Our analysis of the parallel and serial generators as given by Theorems 1 and 2 indicates that their advantage functions depend differently on the advantage function of the underlying primitive F. (We model the latter as a pseudorandom function [13] and associate an advantage function as per [5].) In general, the parallel generator provides better security. This is true already when F is a block cipher but even more strikingly the case when F is a non-invertible PRF. This should be kept in mind when choosing between the generators for re-keying. However, whether or not it eventually helps depends also on the application. For example, with CBC encryption, there is no particular difference in the quantitative security providing by parallel and serial re-keying (even though both provide gains over the single-key scheme). This is due to the shape of the curve of the advantage function of the base CBC encryption function as explained in Section 3.

FORWARD SECURITY. Another possible motivation for re-keying is to provide forward security. The goal here is to minimize the amount of damage that might be caused by key exposure due, for instance, to compromise of the security of the underlying system storing the secret key. (Forward security was first considered for session keys [15,12] and then for digital signatures [7].) Under re-keying, the adversary would only get the current subkey and state of the system. It could certainly figure out all future subkeys, but what about past ones? If the re-keying process is appropriately designed, it can have forward security: the past subkeys will remain computationally infeasible for the adversary to derive even given the current subkey and state, and thus ciphertexts that were formed under them will not be compromised. It is easy to see that the parallel generator does not provide forward security. It can be shown however that the serial one does. A treatment of forward security in the symmetric setting, including a proof of the forward security of the serial generator and the corresponding re-keyed encryption scheme, can be found in [9].

RELATED WORK. Another approach to increasing the encryption threshold, discussed in [6], is to use a mode of encryption not subject to birthday attack (e.g. CTR rather than CBC) and implement this using a non-invertible, high security PRF rather than a block cipher. Constructions of appropriate PRFs have been provided in [6,16]. Re-keying is cheaper in that one can use the given block cipher and a standard mode like CBC and still push the encryption threshold well beyond the birthday threshold.

Re-keying requires that parties maintain state. Stateless methods of increasing security beyond the birthday bound are discussed in [4].

2 Re-keying Processes as Pseudorandom Generators

The subkeys derived by a re-keying process may be used in many different ways: data encryption or authentication are some but not all of these. To enable modular analysis, we separate the subkey generation from the application that uses the subkeys. We view the re-keying process —which generates the subkeys— as a stateful pseudorandom bit generator. In this section we provide quantitative assessments of the security of various re-keying schemes with regard to notions of security for pseudorandom generators. These application independent results are used in later sections to assess the security of a variety of different applications under re-keying.

STATEFUL GENERATORS. A stateful generator $\mathcal{G} = (\mathcal{K}, \mathcal{N})$ is a pair of algorithms. The probabilistic *key generation* algorithm $\mathcal{K}$ produces the initial state, or seed, of the generator. The deterministic *next step* algorithm $\mathcal{N}$ takes the current state as input and returns a block, viewed as the output of this stage, and an updated state, to be stored and used in the next invocation. A sequence $Out_1, Out_2, \ldots$ of pseudorandom blocks is defined by first picking an initial seed $St_0 \leftarrow \mathcal{K}$ and then iterating: $(Out_i, St_i) \leftarrow \mathcal{N}(St_{i-1})$ for $i \geq 1$. (When the generator is used for re-keying, these are the subkeys. Thus Out_i was denoted K_i in Section 1). We assume all output blocks are of the same length and call this the block length.

We now specify two particular generators, the parallel and serial ones. We fix a PRF F: $\{0,1\}^k \times \{0,1\}^k \to \{0,1\}^k$. (As the notation indicates, we are making the simplifying assumption that the key length, as well as the input and output lengths of each individual function $F(K,\cdot)$ are all equal to k.) In practice, this might be instantiated via a block cipher or via a keyed hash function such as HMAC [2]. (For example, if DES is used, then we set $k = 64$ and define $F(K,\cdot)$ to be DES(K[1..56], $\cdot$).)

Construction 1. (Parallel generator) The F-based parallel generator $\mathcal{PG}[F] = (\mathcal{K},\mathcal{N})$ is defined by

```
Algorithm $\mathcal{K}$
$K \overset{R}{\leftarrow} \{0,1\}^k$
Return $\langle 0, K \rangle$
```

```
Algorithm $\mathcal{N}(\langle i, K \rangle)$
$Out \leftarrow F(K, i)$
Return $(Out, \langle i+1, K \rangle)$
```

The state has the form $\langle i, K \rangle$ where K is the initial seed and i is a counter, initially zero. In the i-th stage, the output block is obtained by applying the K-keyed PRF to the (k-bit binary representation of the integer) i, and the counter is updated. This generator has block length k. ▮

Construction 2. (Serial generator) The F-based serial generator $\mathcal{SG}[F] = (\mathcal{K},\mathcal{N})$ is defined by

```
Algorithm $\mathcal{K}$
$K \overset{R}{\leftarrow} \{0,1\}^k$
Return $K$
```

```
Algorithm $\mathcal{N}(K)$
$Out \leftarrow F(K, 0)$
$K \leftarrow F(K, 1)$
Return $(Out, K)$
```

The state is a key K. In the i-th stage, the output block is obtained by applying the K-keyed PRF to the (k-bit binary representation of the integer) 0, and the new state is a key generated by applying the K-keyed PRF to the (k-bit binary representation of the integer) 1. This generator has block length k. ▮

Pseudorandomness. The standard desired attribute of a (stateful) generator is pseudorandomness of the output sequence. We adopt the notion of [11,18] which formalizes this by asking that the output of the generator on a random seed be computationally indistinguishable from a random string of the same length. Below, we concretize this notion by associating to any generator an advantage function which measures the probability that an adversary can detect a deviation in pseudorandomness as a function of the amount of time invested by the adversary.

Definition 1. (Pseudorandomness of a stateful generator) Let $\mathcal{G} = (\mathcal{K},\mathcal{N})$ be a stateful generator with block length k, let n be an integer, and let A be an adversary. Consider the experiments

```
Experiment $\mathbf{Exp}^{\text{prg-real}}_{\mathcal{G},n}(A)$
    $St_0 \leftarrow \mathcal{K}$ ; $s \leftarrow \varepsilon$
    for $i = 1, \ldots, n$ do
        $(Out_i, St_i) \leftarrow \mathcal{N}(St_{i-1})$ ; $s \leftarrow s \,\|\, Out_i$
    $g \leftarrow A(s)$
    return $g$
```

```
Experiment $\mathbf{Exp}^{\text{prg-rand}}_{\mathcal{G},n}(A)$
    $s \leftarrow \{0,1\}^{n \cdot k}$
    $g \leftarrow A(s)$
    return $g$
```

Now define the *advantage* of A and the *advantage function of the generator*, respectively, as follows:

$$\mathsf{Adv}^{\text{prg}}_{\mathcal{G},n}(A) = \Pr[\,\mathbf{Exp}^{\text{prg-real}}_{\mathcal{G},n}(A) = 1\,] - \Pr[\,\mathbf{Exp}^{\text{prg-rand}}_{\mathcal{G},n}(A) = 1\,]$$
$$\mathsf{Adv}^{\text{prg}}_{\mathcal{G},n}(t) = \max_{A}\{\,\mathsf{Adv}^{\text{prg}}_{\mathcal{G},n}(A)\,\}\ ,$$

where the maximum is over all A with "time-complexity" t. ▮

Here "time-complexity" is the maximum of the execution times of the two experiments plus the size of the code for A, all in some fixed RAM model of computation. (Note that the execution time refers to that of the entire experiment, not just the execution time of the adversary.) The advantage function is the maximum likelihood of the security of the pseudorandom generator $\mathcal{G}$ being compromised by an adversary using the indicated resources.

Security measure for PRFs. Since the security of the above constructions depends on that of the underlying PRF F: $\{0,1\}^k \times \{0,1\}^k \rightarrow \{0,1\}^k$, we recall the measure of [5], based on the notion of [13]. Let R^k denote the family of all functions mapping $\{0,1\}^k$ to $\{0,1\}^k$, under the uniform distribution. If D is a distinguisher having an oracle, then

$$\mathsf{Adv}^{\text{prf}}_{F}(D) \;=\; \Pr[\,D^{F(K,\cdot)} = 1 \;:\; K \overset{R}{\leftarrow} \{0,1\}^k\,] - \Pr[\,D^{f(\cdot)} = 1 \;:\; f \overset{R}{\leftarrow} R^k\,]$$

is the advantage of D. The advantage function of F is

$$\mathsf{Adv}^{\text{prf}}_{F}(t,q) \;=\; \max_{D}\{\,\mathsf{Adv}^{\text{prf}}_{F}(D)\,\}\ ,$$

where the maximum is over all A with "time-complexity" t and making at most q oracle queries. The time-complexity is the execution time of the experiment $K \overset{R}{\leftarrow} \{0,1\}^k$; $v \leftarrow D^{F(K,\cdot)}$ plus the size of the code of D, and, in particular, includes the time to compute $F_K(\cdot)$ and reply to oracle queries of D.

Pseudorandomness of the parallel and serial generators. The following theorems, whose proofs can be found in Appendices A and B, show how the pseudorandomness of the two generators is related to the security of the underlying PRF.

Theorem 1. *Let F: $\{0,1\}^k \times \{0,1\}^k \rightarrow \{0,1\}^k$ be a PRF and let $\mathcal{PG}[F]$ be the F-based parallel generator defined in Construction 1. Then*

$$\mathsf{Adv}^{\text{prg}}_{\mathcal{PG}[F],n}(t) \leq \mathsf{Adv}^{\text{prf}}_{F}(t,n)\ . ▮$$

Theorem 2. *Let F: $\{0,1\}^k \times \{0,1\}^k \to \{0,1\}^k$ be a PRF and let $\mathcal{SG}[F]$ be the F-based parallel generator defined in Construction 2. Then*

$$\mathsf{Adv}^{\mathrm{prg}}_{\mathcal{SG}[F],n}(t) \le n \cdot \mathsf{Adv}^{\mathrm{prf}}_{F}(t + \log n, 2)\ . \blacksquare$$

The qualitative interpretation of the two theorems is the same: both the parallel and the serial generator are secure pseudorandom bit generators if the PRF is secure. The quantitative statements show however that the pseudorandomness of n output blocks depends differently on the security of the PRF in the two cases. For the parallel generator, it depends on the security of the PRF under n queries. For the serial generator, it depends on the security of the PRF against only a constant number of queries, but this term is multiplied by the number of output blocks. Comparing the functions on the right hand side in the two theorems will tell us which generator is more secure.

Examples. As an example, assume F is a block cipher. Since F is a cipher, each map $F(K,\cdot)$ is a permutation, and birthday attacks can be used to distinguish F from the family of random functions with a success rate growing as $q^2/2^k$ for q queries (c.f.. [5, Proposition 2.4]). Let us make the (heuristic) assumption that this is roughly the best possible, meaning

$$\mathsf{Adv}^{\mathrm{prf}}_{F}(t,q) \approx \frac{q^2+t}{2^k} \tag{1}$$

for t small enough to prevent cryptanalytic attacks. Now the above tells us that the advantage functions of the two generators grow as follows:

$$\mathsf{Adv}^{\mathrm{prg}}_{\mathcal{PG}[F],n}(t) \approx \frac{n^2+t}{2^k} \text{ and } \mathsf{Adv}^{\mathrm{prg}}_{\mathcal{SG}[F],n}(t) \approx \frac{nt}{2^k}\ .$$

Since $t \ge n$, the two functions are roughly comparable, but in fact the first one has a somewhat slower growth because we would expect that $t \gg n$. So, in this case, the parallel generator is somewhat better.

Now assume F is not a block cipher but something that better approximates a random function, having security beyond the birthday bound. Ideally, we would like something like

$$\mathsf{Adv}^{\mathrm{prf}}_{F}(t,q) \approx \frac{q+t}{2^k} \tag{2}$$

for t small enough to prevent cryptanalytic attacks. This might be achieved by using a keyed hash function based construction, or by using PRFs constructed from block ciphers as per [6,16]. In this case we would get

$$\mathsf{Adv}^{\mathrm{prg}}_{\mathcal{PG}[F],n}(t) \approx \frac{n+t}{2^k} \text{ and } \mathsf{Adv}^{\mathrm{prg}}_{\mathcal{SG}[F],n}(t) \approx \frac{nt}{2^k}\ .$$

Thinking of $t \approx n$ (it cannot be less but could be more, so this is an optimistic choice), we see that the first function has linear growth and the second has quadratic growth, meaning the parallel generator again offers better security, but this time in a more decisive way.

These examples illustrate how the quantitative results of the theorems can be coupled with cryptanalytic knowledge or assumptions about the starting primitive F to yield information enabling a user to choose between the generators.

3 Re-keyed Symmetric Encryption

We fix a *base encryption scheme.* (For example, CBC mode encryption based on some block cipher.) We wish to encrypt data using this scheme, but with re-keying. Two things need to be decided. The first is how the re-keying is to be done, meaning how the subkeys will be computed. This corresponds to making a choice of stateful generator to generate the subkey sequence. The second is the lifetime of each subkey, meaning how many encryptions will be done with it. This corresponds to choosing an integer parameter $l > 0$ which we call the *subkey lifetime.* Associated to a base scheme, generator and subkey lifetime, is a particular *re-keyed encryption scheme.* We are interested in comparing the security of the re-keyed encryption scheme across different choices of re-keying processes (i.e. generators), keeping the base scheme and subkey lifetime fixed. In particular, we want to compare the use of the parallel and serial generators.

Our analysis takes a modular approach. Rather than analyzing separately the re-keyed encryption schemes corresponding to different choices of generators, we first analyze the security of a re-keyed encryption scheme with an arbitrary generator, showing how the advantage of the encryption scheme can be bounded in terms of that of the generator and the base scheme. We then build on results of Section 2 to get results for re-keyed encryption with specific generators. We begin by specifying in more detail the re-keyed encryption scheme and saying how we measure security of symmetric encryption schemes.

Re-keyed encryption schemes. Let $\mathcal{SE} = (\mathcal{K}_e, \mathcal{E}, \mathcal{D})$ be the base (symmetric) encryption scheme, specified by its key generation, encryption and decryption algorithms [3]. Let $\mathcal{G} = (\mathcal{K}_g, \mathcal{N})$ be a stateful generator with block size k, where k is the length of the key of the base scheme. Let $l > 0$ be a subkey lifetime parameter. We associate to them a *re-keyed encryption scheme* $\overline{\mathcal{SE}}[\mathcal{SE}, \mathcal{G}, l] = (\overline{\mathcal{K}}, \overline{\mathcal{E}}, \overline{\mathcal{D}})$. This is a stateful encryption scheme which works as follows. The initial state of the encryption scheme includes the initial state of the generator, given by $St_0 \overset{R}{\leftarrow} \mathcal{K}_g$. Encryption is divided into stages $i = 1, 2, \ldots$. Stage i begins with the generation of a new key K_i using the generator: $(K_i, St_i) \leftarrow \mathcal{N}(St_{i-1})$. In stage i encryption is done using the encryption algorithm of the base scheme with key K_i. An *encryption counter* is maintained, and when l encryptions have been performed, this stage ends. The encryption counter is then reset, the stage counter is incremented, and the key for the next stage is generated. If the base scheme is stateful, its state is reset whenever the key changes.

Formally, the key generation algorithm $\overline{\mathcal{K}}$ of the re-keyed scheme is run once, at the beginning, to produce an initial state which is shared between sender and receiver and includes St_0. The encryption algorithm $\overline{\mathcal{E}}$ takes the current state (which includes K_i, St_i, a stage counter, the encryption counter, and a state for the base scheme if the latter happens to be stateful) and the message M to be encrypted, and returns ciphertext $C \leftarrow \mathcal{E}_{K_i}(M)$. It also returns an updated state which is stored locally. It is advisable to include with the ciphertext the number i of the current stage, so that the receiver can maintain decryption capability even if messages are lost in transit. The $\overline{\mathcal{D}}$ algorithm run by the receiver can be stateless in this case. (This is true as long as the goal is privacy against chosen-plaintext attacks as we consider here, but if active attacks are considered, meaning we want privacy against chosen-ciphertext attacks or authenticity, the receiver will have to maintain state as well.)

Security measures for encryption schemes. Several (polynomial-time equivalent) definitions for security of a symmetric encryption scheme under chosen-plaintext attack were given in [3]. We use one of them, called left-or-right security. The game begins with a random bit b being chosen. The adversary then gets access to an oracle which can take as input any two equal-length messages (x_0, x_1) and responds with a ciphertext formed by encrypting x_b. The adversary wins if it can eventually guess b correctly. We can associate to any adversary an advantage measuring the probability it wins. We then associate to the base encryption scheme —respectively, the re-keyed encryption scheme— an advantage function $\mathsf{Adv}^{\text{ind-cpa}}_{\mathcal{SE}}(t, q, m)$ —respectively $\mathsf{Adv}^{\text{ind-cpa}}_{\overline{\mathcal{SE}}}(t, q, m)$— which measures the maximum probability of the scheme being compromised by an adversary running in time t and allowed q oracle queries each consisting of a pair of m-bit messages. Intuitively, this captures security against a chosen-plaintext attack of q messages. (The usual convention [3] is to allow messages of different lengths and count the sum of the lengths of all messages but for simplicity we ask here that all messages have the same length. Note that for the base encryption scheme, all encryption is done using a single, random key. For the re-keyed scheme, it is done as the scheme specifies, meaning with the key changing every l encryptions. We omit details here, but precise definitions with this type of notation can be found for example in [8].)

Security of re-keyed encryption. The qualitative interpretation of the following theorem is that if the generator and base encryption scheme are secure then so is the re-keyed encryption scheme. It is the quantitative implications however on which we focus. The theorem says that the security of encrypting ln messages with the re-keyed scheme relates to the pseudorandomness of n blocks of the generator output and the security of encrypting l messages under the base scheme with a single random key. The $\mathsf{Adv}^{\text{ind-cpa}}_{\mathcal{SE}}(t, l, m)$ term is multiplied by n, yet there is a clear gain, in that the security of the base encryption scheme relates to encrypting only l messages. The proof of Theorem 3 can be found in the full version of this paper [1].

Theorem 3. (Security of re-keyed encryption) *Let $\mathcal{SE}$ be a base encryption scheme with key size k, let $\mathcal{G}$ be a stateful generator with blocksize k, and let $l > 0$ be a subkey lifetime. Let $\overline{\mathcal{SE}} = \overline{\mathcal{SE}}[\mathcal{SE}, \mathcal{G}, l]$ be the associated re-keyed encryption scheme. Then*

$$\mathsf{Adv}^{\text{ind-cpa}}_{\overline{\mathcal{SE}}}(t, ln, m) \;\leq\; \mathsf{Adv}^{\text{prg}}_{\mathcal{G},n}(t) + n \cdot \mathsf{Adv}^{\text{ind-cpa}}_{\mathcal{SE}}(t, l, m)\ . \blacksquare$$

Re-keyed encryption with the parallel and serial generators. Combining Theorem 3 with Theorems 1 and 2 gives us information about the security of re-keyed encryption under the parallel and serial generators.

Corollary 1. (Security of re-keyed encryption with the parallel generator) *Let $\mathcal{SE}$ be a base encryption scheme, let $F\colon \{0,1\}^k \times \{0,1\}^k \to \{0,1\}^k$ be a PRF, let $\mathcal{PG}[F]$ be the F-based parallel generator defined in Construction 1, and let $l > 0$ be a subkey lifetime. Let $\overline{\mathcal{SE}} = \overline{\mathcal{SE}}[\mathcal{SE}, \mathcal{PG}[F], l]$ be the associated re-keyed encryption scheme. Then*

$$\mathsf{Adv}^{\text{ind-cpa}}_{\overline{\mathcal{SE}}}(t, ln, m) \;\leq\; \mathsf{Adv}^{\text{prf}}_{F}(t, n) + n \cdot \mathsf{Adv}^{\text{ind-cpa}}_{\mathcal{SE}}(t, l, m) \;. \blacksquare$$

Corollary 2. (Security of re-keyed encryption with the serial generator) *Let $\mathcal{SE}$ be a base encryption scheme, let $F\colon \{0,1\}^k \times \{0,1\}^k \to \{0,1\}^k$ be a PRF, let $\mathcal{SG}[F]$ be the F-based serial generator defined in Construction 2, and let $l > 0$ be a subkey lifetime. Let $\overline{\mathcal{SE}} = \overline{\mathcal{SE}}[\mathcal{SE}, \mathcal{SG}[F], l]$ be the associated re-keyed encryption scheme. Then*

$$\mathsf{Adv}^{\text{ind-cpa}}_{\overline{\mathcal{SE}}}(t, ln, m) \;\leq\; n \cdot \mathsf{Adv}^{\text{prf}}_{F}(t + \log n, 2) + n \cdot \mathsf{Adv}^{\text{ind-cpa}}_{\mathcal{SE}}(t, l, m) \;. \blacksquare$$

Example. For the base encryption scheme, let us use CBC with some block cipher $B\colon \{0,1\}^k \times \{0,1\}^b \to \{0,1\}^b$ having block length b. We wish to compare the security of encrypting q messages directly with one key; doing this with re-keying using the parallel generator; and doing this with re-keying using the serial generator. The re-keying is based on a PRF $F\colon \{0,1\}^k \times \{0,1\}^k \to \{0,1\}^k$ having block length k. Note that B and F can but need not be the same. In particular B must be a cipher (i.e. invertible) in order to enable CBC decryption, but we have seen that better security results for the re-keying schemes by choosing F to be non-invertible and might want to choose F accordingly.

Let $\mathcal{CBC}$ denote the base encryption scheme. Let $\mathcal{PCBC}$ denote the re-keyed encryption scheme using $\mathcal{CBC}$ as the base scheme, the F-based parallel generator, and subkey lifetime parameter l. Let $\mathcal{SCBC}$ denote the re-keyed encryption scheme using $\mathcal{CBC}$ as the base scheme, the F-based serial generator, and subkey lifetime parameter l. Since B is a cipher we take its advantage to be

$$\mathsf{Adv}^{\text{prf}}_{B}(t, q) \approx \frac{q^2}{2^b} + \frac{t}{2^k} \;. \tag{3}$$

We know from [3] that

$$\mathsf{Adv}^{\text{ind-cpa}}_{\mathcal{CBC}}(t, q, m) \approx \frac{q^2 m^2}{b^2 2^b} + 2 \cdot \mathsf{Adv}^{\text{prf}}_{B}(t, qm/b) \approx \frac{3q^2 m^2}{b^2 2^b} + \frac{2t}{2^k} \;.$$

For simplicity we let the message length be $m = b$. Thus if $q = ln$ messages of length m are CBC encrypted we have

$$\begin{aligned}
\mathsf{Adv}^{\text{ind-cpa}}_{\mathcal{CBC}}(t, ln, b) &\approx \frac{3l^2 n^2}{2^b} + \frac{2t}{2^k} \\
\mathsf{Adv}^{\text{ind-cpa}}_{\mathcal{PCBC}}(t, ln, b) &\approx \mathsf{Adv}^{\text{prf}}_{F}(t, n) + \frac{3l^2 n}{2^b} + \frac{2nt}{2^k} \\
\mathsf{Adv}^{\text{ind-cpa}}_{\mathcal{SCBC}}(t, ln, b) &\approx n \cdot \mathsf{Adv}^{\text{prf}}_{F}(t + \log n, 2) + \frac{3l^2 n}{2^b} + \frac{2nt}{2^k} \;.
\end{aligned}$$

The first corresponds to encryption with a single key, the second to re-keying with the parallel generator, and the third to re-keying with the serial generator.

Suppose we let F be a block cipher. (This is the easiest choice in practice.) We can simply let $F = B$. In that case F obeys Equation (1) and we get

$$\begin{aligned}\mathsf{Adv}^{\text{ind-cpa}}_{\mathcal{CBC}}(t, ln, m) &\approx \frac{3l^2n^2 + 2t}{2^k}\\ \mathsf{Adv}^{\text{ind-cpa}}_{\mathcal{PCBC}}(t, ln, m) &\approx \frac{3l^2n + n^2 + 2nt}{2^k}\\ \mathsf{Adv}^{\text{ind-cpa}}_{\mathcal{SCBC}}(t, ln, m) &\approx \frac{3l^2n + 2nt + t}{2^k}\ .\end{aligned}$$

The two generators deliver about the same advantage. To gauge the gains provided by the re-keying schemes over the single-key scheme, let us define the *encryption threshold* of a scheme to be the smallest number of messages $Q = ln$ that can be encrypted before the advantage hits one. (Roughly speaking, this is the number of messages we can safely encrypt.) We want it to be as high as possible. Let's take $t \approx nl$. (It cannot be less but could be more so this is an optimistic choice). In the single-key scheme $Q \approx 2^{k/2}$. In the re-keyed schemes let us set $l = 2^{k/3}$. (This is the optimal choice.) In that case $Q \approx 2^{2k/3}$. This is a significant increase in the encryption threshold, showing that re-keying brings important security benefits.

We could try to set F to be a non-invertible PRF for which Equation (2) is true. (In particular F would not be B.) Going through the calculations shows that again the two generators will offer the same advantage, but this would be an improvement over the single-key scheme only if $k > b$. (Setting $k = 2b$ yields an encryption threshold of 2^b for the re-keyed schemes as compared to $2^{b/2}$ for the single-key scheme.)

We saw in Section 2 that the parallel generator offered greater security than the serial one. We note that this did not materialize in the application to re-keyed CBC encryption: here, the advantage functions arising from re-keying under the two generators are the same. This is because the term corresponding to the security of the base scheme in Corollaries 1 and 2 dominates when the base scheme is CBC.

In summary we wish to stress two things: that security increases are possible, and that our results provide general tools to estimate security in a variety of re-keyed schemes and to choose parameters to minimize the advantage functions of the re-keyed schemes.

Acknowledgments

Michel Abdalla was supported in part by CAPES under Grant BEX3019/95-2. Mihir Bellare was supported in part by a 1996 Packard Foundation Fellowship in Science and Engineering and NSF CAREER Award CCR-9624439.

References

1. M. Abdalla and M. Bellare, "A comparative analysis of the security of re-keying techniques," Full version of this paper, available via http://www-cse.ucsd.edu/users/mihir.

2. M. Bellare, R. Canetti and H. Krawczyk, "Keying hash functions for message authentication," *Advances in Cryptology – Crypto '96*, LNCS Vol. 1109, N. Koblitz ed., Springer-Verlag, 1996.
3. M. Bellare, A. Desai, E. Jokipii and P. Rogaway, "A concrete security treatment of symmetric encryption: Analysis of the DES modes of operation," *Proc. of the 38th* IEEE FOCS, IEEE, 1997.
4. M. Bellare, O. Goldreich and H. Krawczyk, "Stateless evaluation of pseudorandom functions: Security beyond the birthday barrier," *Advances in Cryptology – Crypto '99*, LNCS Vol. 1666, M. Wiener ed., Springer-Verlag, 1999.
5. M. Bellare, J. Kilian and P. Rogaway, "The security of cipher block chaining," available via `http://www-cse.ucsd.edu/users/mihir`. Preliminary version in *Advances in Cryptology – Crypto '94*, LNCS Vol. 839, Y. Desmedt ed., Springer-Verlag, 1994.
6. M. Bellare, T. Krovetz and P. Rogaway, "Luby-Rackoff backwards: Increasing security by making block ciphers non-invertible," *Advances in Cryptology – Eurocrypt '98*, LNCS Vol. 1403, K. Nyberg ed., Springer-Verlag, 1998.
7. M. Bellare and S. Miner, "A forward-secure digital signature scheme," *Advances in Cryptology – Crypto '99*, LNCS Vol. 1666, M. Wiener ed., Springer-Verlag, 1999.
8. M. Bellare and C. Namprempre, "Authenticated Encryption: Relations among notions and analysis of the generic composition paradigm," *Advances in Cryptology – ASIACRYPT '00*, LNCS Vol. ??, T. Okamoto ed., Springer-Verlag, 2000. Available via `http://www-cse.ucsd.edu/users/mihir`.
9. M. Bellare and B. Yee, "Forward security in private-key cryptography," Manuscript, 1998.
10. E. Biham and A. Shamir, "Differential cryptanalysis of the Full 16-round DES," *Advances in Cryptology – Crypto '92*, LNCS Vol. 740, E. Brickell ed., Springer-Verlag, 1992.
11. M. Blum and S. Micali, "How to generate cryptographically strong sequences of pseudo-random bits," *SIAM Journal on Computing*, Vol. 13, No. 4, 850-864, November 1984.
12. W. Diffie, P. van Oorschot and M. Wiener, "Authentication and authenticated key exchanges," *Designs, Codes and Cryptography*, 2, 107–125, 1992.
13. O. Goldreich, S. Goldwasser and S. Micali, "How to construct random functions," *Journal of the ACM,* Vol. 33, No. 4, 1986, pp. 210–217.
14. S. Goldwasser and S. Micali, "Probabilistic encryption," *Journal of Computer and System Sciences*, Vol. 28, 1984, pp. 270–299.
15. C. Günther, "An identity-based key-exchange protocol," *Advances in Cryptology – Eurocrypt '89*, LNCS Vol. 434, J-J. Quisquater, J. Vandewille ed., Springer-Verlag, 1989.
16. C. Hall, D. Wagner, J. Kelsey and B. Schneier, "Building PRFs from PRPs," *Advances in Cryptology – Crypto '98*, LNCS Vol. 1462, H. Krawczyk ed., Springer-Verlag, 1998.
17. M. Matsui, "The first experimental cryptanalysis of the Data Encryption Standard," *Advances in Cryptology – Crypto '94*, LNCS Vol. 839, Y. Desmedt ed., Springer-Verlag, 1994.
18. A. Yao, "Theory and applications of trapdoor functions," *Proc. of the 23rd* IEEE FOCS, IEEE, 1982.

A Proof of Theorem 1

Let A be an adversary attacking the pseudorandomness of $\mathcal{PG}[F]$ and t be the maximum of the running times of $\mathbf{Exp}^{\text{prg-real}}_{\mathcal{PG}[F],n}(A)$ and $\mathbf{Exp}^{\text{prg-rand}}_{\mathcal{PG}[F],n}(A)$. We want to upper bound $\mathsf{Adv}^{\text{prg}}_{\mathcal{PG}[F],n}(A)$. We do so by constructing a distinguisher D for F and relating its advantage to that of A. D has access to an oracle $\mathcal{O}$. It simply computes $s = \mathcal{O}(1) \,\|\, \ldots \,\|\, \mathcal{O}(n)$ and outputs the same guess as A on input s. We can see that when the oracle $\mathcal{O}$ is drawn at random from the family F, the probability that D returns 1 equals the probability that the experiment $\mathbf{Exp}^{\text{prg-real}}_{\mathcal{PG}[F],n}(A)$ returns 1. Likewise, the probability that the experiment $\mathbf{Exp}^{\text{prg-rand}}_{\mathcal{PG}[F],n}(A)$ returns 1 equals that of D returning 1 when $\mathcal{O}$ is drawn at random from the family of random functions R^k. As D runs in time at most t and makes exactly n queries to its oracle, we get that

$$\mathsf{Adv}^{\text{prg}}_{\mathcal{PG}[F],n}(A) \;\leq\; \mathsf{Adv}^{\text{prf}}_{F}(t,n)\ .$$

Since A was an arbitrary adversary and the maximum of the running times of experiments $\mathbf{Exp}^{\text{prg-real}}_{\mathcal{PG}[F],n}(A)$ and $\mathbf{Exp}^{\text{prg-rand}}_{\mathcal{PG}[F],n}(A)$ is t, we obtain the conclusion of the theorem.

B Proof of Theorem 2

Let A be an adversary attacking the pseudorandomness of $\mathcal{SG}[F]$ and t be the maximum of the running times of $\mathbf{Exp}^{\text{prg-real}}_{\mathcal{SG}[F],n}(A)$ and $\mathbf{Exp}^{\text{prg-rand}}_{\mathcal{SG}[F],n}(A)$. We want to upper bound $\mathsf{Adv}^{\text{prg}}_{\mathcal{SG}[F],n}(A)$. We begin by defining the following sequence of hybrid experiments, where j varies between 0 and n.

```
Experiment $\mathbf{Hybrid}(A, j)$
    $St \xleftarrow{R} \{0,1\}^k$ ; $s \leftarrow \varepsilon$
    for $i = 1, \ldots, n$ do
        if $i \leq j$ then $Out_i \xleftarrow{R} \{0,1\}^k$
        else $(Out_i, St) \leftarrow \mathcal{N}(St)$
        $s \leftarrow s \,\|\, Out_i$
    $g \leftarrow A(s)$
    return $g$
```

Let P_j be the probability that experiment $\mathbf{Hybrid}(A, j)$ returns 1, for $j = 0, \ldots, n$. Note that the experiments $\mathbf{Exp}^{\text{prg-real}}_{\mathcal{SG}[F],n}(A)$ and $\mathbf{Exp}^{\text{prg-rand}}_{\mathcal{SG}[F],n}(A)$ are identical to $\mathbf{Hybrid}(A, 0)$ and $\mathbf{Hybrid}(A, n)$, respectively. (Not syntactically, but semantically.) This means that $P_0 = \Pr[\,\mathbf{Exp}^{\text{prg-real}}_{\mathcal{SG}[F],n}(A) = 1\,]$ and $P_n = \Pr[\,\mathbf{Exp}^{\text{prg-rand}}_{\mathcal{SG}[F],n}(A) = 1\,]$. Putting it all together, we have

$$\begin{aligned}\mathsf{Adv}^{\text{prg}}_{\mathcal{SG}[F],n}(A) &= \Pr[\,\mathbf{Exp}^{\text{prg-real}}_{\mathcal{SG}[F],n}(A) = 1\,] - \Pr[\,\mathbf{Exp}^{\text{prg-rand}}_{\mathcal{SG}[F],n}(A) = 1\,] \\ &= P_0 - P_n\ . \qquad (4)\end{aligned}$$

We now claim that

$$\mathsf{Adv}^{\mathrm{prg}}_{\mathcal{SG}[F],n}(A) = P_0 - P_n \leq \ n \cdot \mathsf{Adv}^{\mathrm{prf}}_{F}(t + \log n, 2)\ . \tag{5}$$

Since A was an arbitrary adversary, we obtain the conclusion of the theorem. It remains to justify Equation (5). We will do this using the advantage function of F. Consider the following distinguisher for F.

```
Algorithm $D^{\mathcal{O}}$
  $j \overset{R}{\leftarrow} \{1,\ldots,n\}$ ; $s \leftarrow \varepsilon$
  for $i = 1,\ldots,n$ do
    if $i < j$ then $Out_i \overset{R}{\leftarrow} \{0,1\}^k$
    if $i = j$ then $Out_i \leftarrow \mathcal{O}(0)$ ; $St \leftarrow \mathcal{O}(1)$
    if $i > j$ then $(Out_i, St) \leftarrow \mathcal{N}(St)$
    $s \leftarrow s \,\|\, Out_i$
  $g \leftarrow A(s)$
  return $g$
```

Suppose the oracle given to D was drawn at random from the family F. Then, the probability that it returns 1 equals the probability that the expirement **Hybrid**$(A, j-1)$ returns 1, where j is the value chosen at random by D in its first step. Similarly, if the given oracle is drawn at random from the family of random functions R^k, then the probability that D returns 1 equals the probability that the experiment **Hybrid**(A, j) returns 1, where j is the value chosen at random by D in its first step. Hence,

$$\begin{aligned} \Pr\left[\, D^{\mathcal{O}} \mid \mathcal{O} \overset{R}{\leftarrow} F \,\right] &= \tfrac{1}{n}\textstyle\sum_{j=1}^{n} P_{j-1} \\ \Pr\left[\, D^{\mathcal{O}} \mid \mathcal{O} \overset{R}{\leftarrow} R^k \,\right] &= \tfrac{1}{n}\textstyle\sum_{j=1}^{n} P_{j}\ . \end{aligned}$$

Subtract the second sum from the first and exploit the collapse to get

$$\frac{P_0 - P_n}{n} \ = \ \tfrac{1}{n}\textstyle\sum_{j=1}^{n} P_{j-1} \ - \ \tfrac{1}{n}\textstyle\sum_{j=1}^{n} P_{j} \ = \ \mathsf{Adv}^{\mathrm{prf}}_{F}(D)\ .$$

Note that D runs in time at most $t + O(\log n)$ and makes exactly 2 queries to its oracle, whence we get Equation (5). This concludes the proof of the theorem.