Towards Fresh and Hybrid Re-Keying Schemes with Beyond Birthday Security

Christoph Dobraunig[1], François Koeune[2], Stefan Mangard[1], Florian Mendel[1], and François-Xavier Standaert[2].

[1] IAIK, Graz University of Technology, Austria.
[2] Université catholique de Louvain – ICTEAM – Crypto Group, Belgium.
christoph.dobraunig@iaik.tugraz.at, francois.koeune@uclouvain.be, stefan.mangard@tugraz.at, florian.mendel@iaik.tugraz.at,fstandae@uclouvain.be

Abstract. Fresh re-keying is a type of protocol which aims at splitting the task of protecting an encryption/authentication scheme against side-channel attacks in two parts. One part, a re-keying function, has to satisfy a minimum set of properties (such as good diffusion), and is based on an algebraic structure that is easy to protect against side-channel attacks with countermeasures such as masking. The other part, a block cipher, brings resistance against mathematical cryptanalysis, and only has to be secure against single-measurement attacks. Since fresh re-keying schemes are cheap and stateless, they are convenient to use in practice and do not require any synchronization between communication parties. However, it has been shown that their first instantiation (from Africacrypt 2010) only provides birthday security because of a (mathematical only) collision-based key recovery attack recently put forward by Dobraunig et al. (CARDIS 2014). In this paper, we provide two provably secure (in the ideal cipher model) solutions to avoid such collision attacks. The first one is based on classical block ciphers, but does not achieve beyond-birthday CPA security (i.e. it only provably prevents the CARDIS 2014 key recovery attack) and requires an additional block cipher execution in the protocol. The second one is based on tweakable block ciphers and provides tight CPA security while also being more efficient. As a complement, we also show that our reasoning extends to hybrid schemes, where the communication party to protect against side-channel attacks is stateful. We illustrate this claim by describing a collision attack against an example of a hybrid scheme patented by Kocher, and presenting a tweak leading to beyond birthday security. We conclude the paper by discussing the use of fresh/hybrid re-keying for encryption and authentication, together with a cautionary note on their side-channel resistance.

1 Introduction

Designing sound and efficient countermeasures against side-channel attacks is a challenging problem. This is especially true in the context of applications with strong cost or energy constraints (e.g. RFIDs, sensor networks, pay-TV, automotive, ...). In such cases, and despite the fact that the devices are likely to be operated in a hostile environment, the direct protection of (e.g.) standard block ciphers such as the AES may be

too expensive. As an illustration, the implementation of the well-known masking countermeasure for such block ciphers implies performance overheads that are (at least) quadratic in the security order [11]. Consequently, a new research path has emerged, trying to reduce the adversary's capabilities thanks to re-keying. Leakage-resilient cryptography is the most investigated representative of this trend (see, e.g. [10, 18]). But unfortunately, the concrete guarantees provided by such constructions highly depend on the primitives. For stateful stream ciphers, leakage-resilience indeed delivers strong security levels at low cost. By contrast, for stateless PRFs and PRPs, the situation is less conclusive (essentially due to the fact that the latter primitives bound the number of plaintexts that an adversary can observe, rather than the number of measurements, and hence allow averaging to get noise-free measurements) [3]. Since stateless primitives are essential ingredients for the initialization of an encryption scheme, or for authentication, we are therefore left with the problem of finding good protection mechanisms in this case.

The fresh re-keying scheme proposed in [16] is a typical attempt in this direction. Here, the authors start from the observation that requiring both physical and mathematical security from a single primitive may be too challenging. Therefore, they suggest an alternative solution, where a stateless re-keying function that only has to fulfill a limited number of mathematical properties and is easy to mask is combined with a mathematically strong block cipher. In this context, hardware engineers essentially have to ensure resistance against multi-trace side-channel attacks (aka DPA resistance) for the re-keying function, and resistance against single-trace side-channel attacks (aka SPA resistance) for the block cipher – the latter being an arguably easier task. While such a construction was indeed interesting from a side-channel attack point-of-view, a recent analysis by Dobraunig et al. showed that such a fresh re-keying scheme only provides birthday security against a (mathematical only) chosen-plaintext collision-based key recovery attack [9]. In this paper, we are therefore interested in improved re-keying mechanisms that provide beyond birthday security.

Our contributions. We start by describing two new re-keying schemes – one fresh and one hybrid – with beyond birthday security against the CARDIS 2014 attack. By hybrid, we mean that one communicating party acts like in a stateful scheme, i.e. the fresh key is based on a secret internal state that is continuously updated, while the other communicating party acts stateless, i.e. the session key is always regenerated from the main secret, based on an index value communicated by the first party. In this way, the first party can be protected against side-channel analysis (as in fresh re-keying), without requiring the synchronization burden of fully stateful schemes (e.g. based on a leakage-resilient PRG). For the first scheme, we rely on a provably secure re-keying proposed by Abdalla and Bellare [1], which allows us to prove the security of our (block cipher or hash function based) re-keying in the ideal cipher model, although the bound is quite loose and does not provide beyond-birthday CPA security (i.e. it only provably prevents the CARDIS 2014 attack). For the second one, we take advantage of tweakable block ciphers to design a very efficient solution, which additionally brings CPA security and benefits from a tight security bound, assuming the existence of an ideal tweakable block cipher. We also suggest concrete instantiations for the building blocks of these schemes, including a couple of new

and very efficient tweakable block ciphers based on the TWEAKEY framework [12], proposed in the context of the ongoing CAESAR competition (e.g. Deoxys, Jotlik, KIASU, and Scream) [8].

We complement these new designs with three additional contributions. First, we put forward that a similar reasoning applies to a hybrid re-keying scheme patented by Kocher [13]. That is, such a scheme is also vulnerable to collision attacks (hence only provides birthday security), and can be fixed by taking advantage of tweakable block ciphers. Second, we discuss the use of fresh/hybrid re-keying schemes in concrete applications, and underline important differences between encryption and authentication in this respect. Eventually, we conclude the paper by recalling the side-channel security guarantees of all the proposed re-keying schemes, including their grey areas regarding the interaction between the re-keying function and its underlying block cipher.

Note that resistance against fault attacks is not discussed in this paper, although all the proposed solutions inherit from the good properties of the original Africacrypt re-keying in this respect, and therefore can probably be used to rule out *differential* fault analysis (such as [5] and following works). As in [16], simple fault attacks (e.g. reducing the number of rounds) are considered out of scope, and have to be prevented by other means.

2 Background

2.1 The Africacrypt 2010 fresh re-keying scheme

The Africarypt 2010 fresh re-keying scheme [16] is pictured in Fig. 1. It is built from a block cipher BC and a re-keying function g, and essentially works in two steps. First, a session key k^* is produced by running the re-keying function on the master key k and a random nonce r (selected uniformly at random by the chip needing to be protected). Second this fresh key k^* is used to encrypt a (single) plaintext x with the block cipher. Note that this scheme is trivially tweaked into a hybrid re-keying by replacing the random nonce r by a counter.

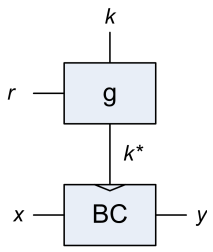


Fig. 1. Africacrypt 2010 fresh re-keying.

2.2 Properties of the g function

Medwed et al. [16] use g to relax the side-channel protection requirements for a block cipher. Informally, the idea is that g will be in charge of generating one-time session keys in a way resistant against side-channel attacks, whereas the block cipher will provide resistance against classical cryptanalysis, but without the need to worry about DPA, as each key is used only once. Since we will re-use the same "separation of duties" strategy and the same g function in the present paper, it is worth recalling the requirements for this function:

1. **Diffusion.** One bit of k^* shall depend on many bits of k.
2. **Stateless Communication.** The parties shall not have an inner state, which has to be kept synchronous.
3. **No additional key material.** k and k^* should have the same size.
4. **Little hardware overhead and side-channel security.** The overhead caused by the use of g (implemented in a secure way) should be small compared to fully protecting the underlying block cipher against side-channel attacks. In other words, the structure of g should make it significantly easier to protect against these attacks (e.g. via masking).
5. **Regularity.** g should have high regularity to facilitate its implementation in a full-custom design (or additional protection mechanisms).

2.3 The CARDIS 2014 collision attack

In this section we describe the attack presented at CARDIS 2014 [9] against the Africacrypt 2010 scheme of Section 2.1. We assume that the generated session keys are used to key a single block cipher encryption. This encryption is used for example in a challenge–response protocol, where the attacker is able to select the challenge x and sees the response $y_i = \mathsf{BC}_{k_i^*}(x)$. The attack can be split into two steps. The first one is the recovery of one session key k_i^*, the second one is the recovery of the master key k out of this knowledge.

The first step is independent of the generation of the session key and targets a single block encryption. In this step the attacker precalculates a list, where he stores pairs of responses (ciphertexts) y_i's and keys k_i^*'s. Those y_i's are encryptions of always the same challenge (plaintext) X for different keys k_i^*. Note that for the creation of the list, all k_i^*'s are chosen by the attacker. Next, in the online phase, the attacker queries an oracle (his target) for multiple encryptions y_i's of the same plaintext X. Since this oracle uses a fresh re-keying scheme, X is encrypted with different keys k_i^*'s and therefore, y_i varies. If such a y_i matches with a y_i in the precalculated list, the corresponding session key is recovered with high probability. For this attack, the best overall complexity of $2 \cdot 2^{n/2}$ is obtained if a list with $2^{n/2}$ entries is used and $2^{n/2}$ online queries are made (for n-bit session keys).

The second step of the attack depends on the concrete re-keying scheme. In the case of the Africacrypt 2010 proposal, g is a multiplication in a polynomial ring. Since we know one session key k^*, and the corresponding nonce r is invertible with a high probability, we can calculate $k = r^{-1} \cdot k^*$.

3 How to do it right?

A natural approach to prevent the CARDIS 2014 attack would be to change the instantiation of the g function and to make it non-invertible. For example, one could use a cryptographic hash function for this purpose. Unfortunately, cryptographic hash functions are not easy to protect against side-channel analysis. In order to circumvent this problem, and as already mentioned, we will use the same "separation of duties" strategy as in the Africacrypt re-keying. That is, we will try to separate the burden of side-channel protection from protection against classical cryptanalysis, but this time including collision attacks in our concerns. For this purpose, we present a fresh/hybrid scheme based on a pseudo-random function (PRF) in Section 3.1, and propose an instantiation thereof that is provably secure in the ideal cipher model. We then propose a more efficient solution based on tweakable block ciphers in Section 3.2.

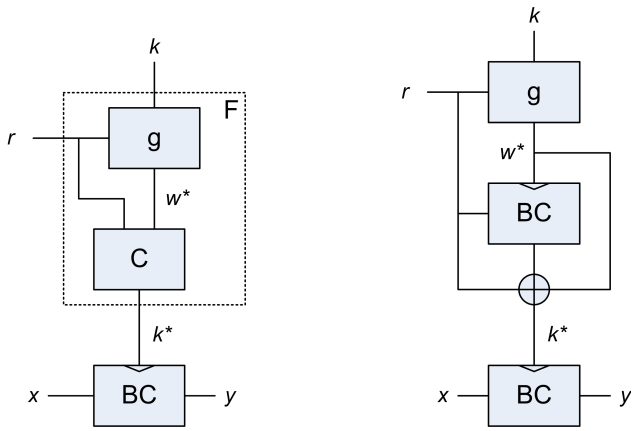
The scenario we focus on in this paper is the case where one communicating party (e.g. the tag) needs re-keying as an easy and cheap protection against side-channel attacks, whereas the other (e.g. the reader) is less cost-sensitive and can be protected through other mechanisms. The re-keying nonce r will thus be randomly chosen by the cheap device and transmitted to the other party. In [15], Mewed et al. considered the scenario where multiple parties must be protected by re-keying and must thus all participate in the selection of r. Their techniques can be straightforwardly applied to our schemes.

3.1 Fresh/hybrid re-keying from the Abdalla-Bellare re-keying

In [1], Abdalla and Bellare proved the security of a PRF-based re-keying scheme. They further suggest to instantiate their PRF with a hash function. Interestingly, such a solution is quite directly applicable in our context. We just need to prove that the combination of g with a well-chosen compression function C is a PRF (represented by the dotted line in Fig. 2a). As previously, the function g shall carry the main burden regarding side-channel protection, whereas the compression function shall prevent collision attacks. Fig. 2a can thus be seen as an extension of the Africacrypt 2010 scheme. Note that here again, the scheme will be stateless if r is a random nonce, and hybrid if it is a counter.

Concretely, and since re-keying schemes are typically combined with block ciphers, we are naturally interested in block cipher-based compression functions. For this purpose, we will analyze one particular construction for C, referred to as the compression function 6 in [6], which Black et al. proved to be pre-image and collision resistant in the ideal cipher model. Such a solution is pictured in Fig. 2b. However, we note that our construction would keep its security properties with any reasonable instantiation of the PRF in Fig. 2a.

We now prove that, in the ideal cipher model, the generation of k^* is a PRF provided g meets a simple requirement, namely that, for any fixed value K of its first parameter, the function $\mathsf{g}(K, \cdot)$ is one-to-one and, for any fixed value R of its second parameter, the function $\mathsf{g}(\cdot, R)$ is one-to-one (in other words, $\mathsf{g}(K, \cdot)$ and $\mathsf{g}(\cdot, R)$ are permutations of the nonce and key space, respectively). Note that the following result is independent on whether r is based on fresh nonces or a counter.



(a) General concept (b) Block cipher based

Fig. 2. Fresh/hybrid Abdalla–Bellare re-keying.

Theorem 1. *Let us define* $\mathsf{F}(k,r) := \mathsf{BC}_{\mathsf{g}(k,r)}(r) \oplus r \oplus \mathsf{g}(k,r)$. *If* $\mathsf{g}(K,\cdot)$ *and* $\mathsf{g}(\cdot,R)$ *are one-to-one for all values of* K, R, *then the construction* F *is a PRF in the ideal cipher model.*

Proof (sketch). Consider an adversary $\mathcal{A}$ trying to distinguish $\mathsf{F}(k,\cdot)$ from a random function. $\mathcal{A}$ receives access to an oracle $\mathcal{O}$ corresponding to the function to test and to an oracle $\mathcal{O}'$ corresponding to the block cipher. When $\mathcal{A}$ starts, $\mathcal{O}$ tosses a coin to decide how it will behave. Depending on the result, on input r_i, $\mathcal{O}$ will:

- either output a fresh, random y_i;
- or output $y_i = \mathsf{BC}_{\mathsf{g}(k,r_i)}(r_i) \oplus r_i \oplus \mathsf{g}(k,r_i)$, for a fixed value of k. In this case, $\mathcal{O}$ in turn obtains the values $\mathsf{BC}_{\mathsf{g}(k,r_i)}(r_i)$ by querying $\mathcal{O}'$.[3]

In addition, $\mathcal{A}$ can directly query $\mathcal{O}'$ with input (k'_i, r'_i) to obtain $\mathsf{BC}_{k'_i}(r'_i)$ or $\mathsf{BC}^{-1}_{k'_i}(r'_i)$.

Consider the case where $\mathcal{O}$ implements the BC-based construction. When $\mathcal{A}$ ends, $\mathcal{O}'$ will thus have received two sets of (possibly intertwined) queries:

- Queries $\mathsf{BC}_{\mathsf{g}(k,r_i)}(r_i)$, through queries to $\mathcal{O}$: we will denote these queries and the corresponding answers as $(r_i, y_i)_{1\le i\le v}$.
- Queries $\mathsf{BC}_{k'_i}(r'_i)$ or $\mathsf{BC}^{-1}_{k'_i}(r'_i)$, through direct calls: we will denote these queries and the corresponding answers as $(k'_i, r'_i, b'_i, y'_i)_{1\le i\le v'}$, where b'_i is a bit equal to 0 (resp. 1) if the request is an encryption (resp. decryption) query.

The central point of the proof is that two queries to $\mathcal{O}'$ yield randomly and independently chosen answers provided the corresponding keys are different. We will now show that this is indeed the case, except with negligible probability.

[3] As usual, we assume that both $\mathcal{O}$ and $\mathcal{O}'$ act consistently: when receiving an input corresponding to a previous query, they simply replay the previous output (when $\mathcal{O}$ implements F, its consistency is a direct consequence of the consistency of $\mathcal{O}'$).

1. If $r_i = r_j$, then it is easy to see that $y_i = y_j$ (the key is the same, but so is the plaintext too, and $\mathcal{O}'$ receives twice the same query).
2. If $r_i \neq r_j$, then, since $\mathsf{g}(k, \cdot)$ is one-to-one, $\mathsf{g}(k, r_i) \neq \mathsf{g}(k, r_j)$, as requested.
3. Since $\mathsf{g}(\cdot, r_i)$ is one-to-one, k is unknown to $\mathcal{A}$, and $\mathcal{A}$ issues only a polynomial number of requests, the probability to have $k'_j = \mathsf{g}(k, r_i)$ for some (i, j) is negligible (no value of r_i allows reducing the destination space of $\mathsf{g}(k, r_i)$).

Summarizing, if $\mathcal{O}$ implements F, then the computation of all (fresh) output values involves an XOR with $\mathsf{BC}_{\mathsf{g}(k,r_i)}(r_i)$, which, except with negligible probability, are chosen randomly and independently by $\mathcal{O}'$. On the other hand, if $\mathcal{O}$ implements a real random function, then all (fresh) output values are chosen randomly and independently. So, in all cases, all answers received by $\mathcal{A}$ are chosen randomly and independently, except with negligible probability, and none of them allows distinguishing F from a random function.

□

Remarks:

1. The intuition behind this proof is that, without knowledge of k, $\mathcal{A}$ cannot query $\mathcal{O}'$ with keys corresponding to one of the values w^* actually used in the scheme, so that its access to $\mathcal{O}'$ does not help $\mathcal{A}$. Note that the possibility to query $\mathcal{O}'$ with different, but related, keys is not ruled out by the structure of the function g (there could for example be a known difference between $\mathsf{g}(k, r_1)$ and $\mathsf{g}(k, r_2)$, no matter the value of k). However, this is not a problem in the ideal cipher model, where related-key attacks do not apply.
2. $\mathcal{A}$ can of course issue direct queries $(\mathsf{BC}_{k'_i}(r'_{i_1}), \mathsf{BC}_{k'_i}(r'_{i_2}))$, which will not yield independent answers. However, these will obviously not reveal any information on F, since, as shown above, they are unrelated to any query made to F.
3. It is worth noting that the properties we require from g are also in line with a work by Bellare and Kohno. In [4], they propose a formal treatment of related-key attacks by providing the adversary with the ability to issue related-key queries such as $E_{\phi(K)}(m)$, i.e. obtain encryptions with a function ϕ of the (unknown) target key, for a carefully defined set Φ of allowed functions. Bellare and Kohno provide some conditions on the set Φ allowing proving resistance against related-key attacks. Interestingly, our construction can be related to theirs by defining $\phi_i(k) = \mathsf{g}(k, r_i)$, and, with the aforementioned conditions on g, match very well the bounds of [4, Def. 2, Def. 3, Lemma 1]. As our construction is slightly different, this paper provides independent proofs. Nevertheless, the fact that our re-keying function matches independently-defined conditions to avoid related-key attacks is a probable witness of the consistency of our approach.

Being able to prove that the re-keying scheme is a PRF is already of interest regarding the CARDIS 2014 collision attack. As a matter of fact, it guarantees that an attacker cannot distinguish the output of F from a random sequence, which of course also implies that he cannot recover the key k that generated this output. As a consequence, this construction is provably resistant against the collision-based key recovery attack in [9].

In addition, Abdalla and Bellare proved in [1, Theorems 1 and 3] that, if F is a PRF, $\mathcal{SE}$ is an encryption scheme and $\overline{\mathcal{SE}}$ is the associated F-based re-keyed encryption scheme, then the advantage of an adversary trying to break $\overline{\mathcal{SE}}$ can be related to that of adversaries trying to break F and $\mathcal{SE}$ as follows:

$$\mathsf{Adv}^{\mathsf{ind-cpa}}_{\overline{\mathcal{SE}}}(t, lm) \leq \mathsf{Adv}^{\mathsf{prf}}_{\mathsf{F}}(t, m) + m \cdot \mathsf{Adv}^{\mathsf{ind-cpa}}_{\mathcal{SE}}(t, l),$$

where t is the adversary's maximum running time, m is the maximum number of keys generated, and l is the maximum number of encryptions performed with each key (so $l = 1$ if we use each fresh key only once).

Unfortunately, this theoretical bound does not provide beyond birthday CPA security. As a matter of fact, an adversary trying $2^{n/2}$ keys against one single block will succeed with probability $2^{-n/2}$, so $\mathsf{Adv}^{\mathsf{ind-cpa}}_{\mathcal{SE}}(2^{n/2}, 1) \geq 2^{-n/2}$. As a consequence, the above bound for an adversary issuing $m = 2^{n/2}$ queries and having computing time $t = 2^{n/2}$ yields:

$$\mathsf{Adv}^{\mathsf{ind-cpa}}_{\overline{\mathcal{SE}}}(2^{n/2}, 2^{n/2}) \leq 1.$$

Interestingly, this bound is tight, since it corresponds to an attack similar to the CARDIS 2014 one, but breaking the CPA game rather than recovering the key. Combined with the observation that the scheme of Figure 2b is also more expensive than the original fresh re-keying scheme, this motivates us to investigate how to overcome these drawbacks, using tweakable block ciphers.

3.2 More efficient solution based on a tweakable block cipher

The scheme presented in Section 3.1 has quite a large performance overhead because of the additional compression/block cipher call needed for a single encryption. A more efficient construction is to replace this combination by a tweakable block cipher TBC, as shown in Fig. 3. Tweakable block ciphers were introduced by Liskov et al. in [14] as a generalized version of block ciphers. In addition to the secret key, a tweakable block cipher accepts a second parameter (that can be public) called the tweak. Intuitively, "*each fixed setting of the tweak gives rise to a different, apparently independent, family of standard block cipher encryption operators*" and a tweakable block cipher should remain secure even facing an adversary who has control of the tweak.

In our context, the publicly known nonce (or counter) r is used as tweak, and k^* as secret key.

This construction again follows the same separation of duties idea as the ones of Fig. 2a and 2b. Namely, the function g is responsible for side-channel protection, whereas the tweakable block cipher prevents the CARDIS 2014 collision attack. Therefore, the requirements for the function g stay the same.

Let us first argue why this construction defeats the aforementioned attacks. We then show it is in fact provably secure in the ideal cipher model.

First recall that the attacks on the re-keying scheme of Africacrypt 2010 exploit the fact that the same plaintext is encrypted with different session keys by the same block cipher. Let us now take into account the fact that $\mathsf{g}(k, \cdot)$ is a permutation. Thus, we get always a different session key k^* for different nonces r. Let us also assume that

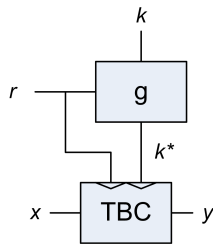


Fig. 3. Fresh/hybrid re–keying with a tweakable block cipher.

we have a perfect tweakable block cipher. This means that for every different value of the tweak, we have different and independent block cipher instances. So, basically, we now just use different block ciphers with different keys, and none of them is used with multiple keys, which makes the CARDIS 2014 attack impossible to apply. Taking another viewpoint, an attacker trying to perform the first step of the attack and precalculating a list would now need to do it, not for a set of k_i^*, but for a set of pairs (r_i, k_i^*). This considerably increases the size of the list before the birthday paradox provides a non-negligible chance of success (since this pair has $2n$-bit size).

If we translate this "perfect TBC" assumption in the ideal cipher model, it simply means that each different values of the key *or* the tweak yields a different, independent permutation. We now prove that, in this model, the construction depicted on Fig. 3 is indeed a TBC (here too, note that the result is independent on whether r is based on fresh nonces or a counter).

Theorem 2. *Let* TBC *be an ideal tweakable block cipher, and let us define* TBC' *as* $\mathsf{TBC}'_k(r,m) = \mathsf{TBC}_{\mathsf{g}(k,r)}(r,m)$*. If* $\mathsf{g}(K,\cdot)$ *and* $\mathsf{g}(\cdot,R)$ *are one-to-one for all values of* K*,* R*, then* TBC' *is a tweakable block cipher.*

Proof (sketch). Consider a distinguisher $\mathcal{D}$ trying to distinguish TBC'_k from a family of independent random permutations. At the beginning of the experiment, an oracle $\mathcal{O}$ tosses a coin to decide which construction it will implement.

- In the first case, $\mathcal{O}$ chooses a random key k and sets $E(r,m) := \mathsf{TBC}_{\mathsf{g}(k,r)}(r,m)$ and $E^{-1}(r,m) := \mathsf{TBC}^{-1}_{\mathsf{g}(k,r)}(r,m)$. In this case, $\mathcal{O}$ in turn obtains these values by querying an oracle $\mathcal{O}'$ implementing the ideal tweakable block cipher.
- In the second case, $\mathcal{O}$ chooses a family $\Pi(\cdot,\cdot)$ of independent random permutations[4] and sets $E(r,m) := \Pi(r,m)$ and $E^{-1}(r,m) := \Pi^{-1}(r,m)$.

$\mathcal{D}$ can then query $\mathcal{O}$ to obtain $E(r_i,m_i)$ or $E^{-1}(r_i,m_i)$ for values (r_i,m_i) of its choice. In addition, $\mathcal{D}$ can also directly query $\mathcal{O}'$ to obtain $\mathsf{TBC}_{k'_i}(r'_i,m'_i)$ or $\mathsf{TBC}^{-1}_{k'_i}(r'_i,m'_i)$ for values (k'_i,r'_i,m'_i) of its choice. The goal of $\mathcal{D}$ is to discover which construction $\mathcal{O}$ implements. We will denote the maximum number of (direct or indirect) queries $\mathcal{D}$ makes to $\mathcal{O}'$ as l.

[4] That is, for each T, $\Pi(T,\cdot)$ is a random permutation of the message space.

Consider the case where $\mathcal{O}$ implements the TBC-based construction. When $\mathcal{D}$ ends, $\mathcal{O}'$ will thus have received two sets of (possibly intertwined) queries:

- Queries $\mathsf{TBC}_{\mathsf{g}(k,r_i)}(r_i, m_i)$ or $\mathsf{TBC}^{-1}_{\mathsf{g}(k,r_i)}(r_i, m_i)$, through queries to $\mathcal{O}$: we will denote these queries and the corresponding answers as $(r_i, m_i, b_i, y_i)_{1\leq i\leq v}$, where b_i is a bit equal to 0 (resp. 1) if the request is an encryption (resp. decryption) query.
- Queries $\mathsf{TBC}_{k'_i}(r'_i, m'_i)$ or $\mathsf{TBC}^{-1}_{k'_i}(r'_i, m'_i)$, through direct calls: we will denote these queries and the corresponding answers as $(k'_i, r'_i, m'_i, b'_i, y'_i)_{1\leq i\leq v'}$, where b'_i is a bit equal to 0 (resp. 1) if the request is an encryption (resp. decryption) query.

Observe that:

1. If $r_i \neq r_j$ (resp. $r'_i \neq r'_j$ and/or $k'_i \neq k'_j$), then y_i and y_j (resp. y'_i and y'_j) are chosen randomly and independently by $\mathcal{O}'$.
2. The same is true when $r_i \neq r'_j$: $\mathcal{O}'$ is queried on different tweak values and provides independent random answers.
3. If $r_i = r_j$, then y_i and y_j are chosen randomly and independently, except that the permutation rule (i.e. different inputs yield different outputs) and consistency rule (i.e. $E(E^{-1}(r, m)) = m$) will be respected. Since Π is also a permutation, none of these limitations helps $\mathcal{D}$ guessing the construction implemented by $\mathcal{O}$.
4. The same argument applies if $(k'_i, r'_i) = (k'_j, r'_j)$.
5. Finally, if $r_i = r'_j$, then, since k is unknown and $\mathsf{g}(\cdot, r_i)$ is one-to-one, the probability to have $k'_j = g(k, r_i)$ is only $\frac{1}{2^n}$. In all other cases, $\mathcal{O}'$ is queried on different key values and provides independent random answers. Since $\mathcal{O}'$ received a maximum of l queries, the global probability is bounded by $\frac{l}{2^n}$.

It is easy to see that, in the case where $\mathcal{O}$ implements a family of independent random permutations, $\mathcal{O}$ will bear exactly the same behavior, except in one case. This only exception is that there is no corresponding to case 5 above ($k'_j = \mathsf{g}(k, r_i)$). This difference of behavior would help discovering the behaviour of $\mathcal{O}$, but, as argued above, only occurs with probability $\frac{l}{2^n}$. In all other cases, answers received by $\mathcal{D}$ are always random, independent values consistent with a permutation.

The distinguishing advantage is thus bounded by:

$$\Pr[D^{\Pi}(t, l) = 1] - \Pr[D^{\mathsf{TBC}'}(t, l) = 1] \leq \frac{l}{2^n}.$$

□

Having proved that our construction is a TBC, we can for example easily prove that encryption based on it is secure against chosen-plaintext attacks.

Theorem 3. *Let* TBC *be a tweakable block cipher and define* $\Pi =< G, E, D >$*, where* $E_k(m)$ *is performed by choosing a random* r *and returning* $E_k(m) := (r, \mathsf{TBC}_k(r, m))$*. In the ideal tweakable cipher model,* Π *provides indistinguishable encryption against a chosen-plaintext adversary.*

Proof (sketch). Consider an adversary $\mathcal{A}$ attacking Π. During the oracle query phases (both before and after the challenge phase), $\mathcal{A}$ can query an encryption oracle $\mathcal{O}$ to obtain the encryption $(r'_i, \mathsf{TBC}_k(r'_i, m'_i))$ of messages he chooses[5], under an unknown

[5] The tweak r'_i, however, is randomly chosen by $\mathcal{O}$.

key k chosen uniformly at random. During the challenge phase, $\mathcal{A}$ outputs two messages m_0, m_1 and receives $c = (r, \mathsf{TBC}_k(r, m_b))$ from $\mathcal{O}$. His goal is to discover b. In both cases, $\mathcal{O}$ answers by querying a TBC oracle $\mathcal{O}'$. $\mathcal{A}$ can also directly query $\mathcal{O}'$ with values (m_i'', k_i'', r_i'') of its choice and obtain $E'_{k_i''}(r_i'', m_i'')$ or $E'^{-1}_{k_i''}(r_i'', m_i'')$. We will denote by l the maximum number of queries (both to $\mathcal{O}$ and $\mathcal{O}'$) issued by $\mathcal{A}$.

Observe that:

- As k was chosen uniformly at random, the probability to have $k = k_i''$ for some i is bounded by $\frac{l}{2^n}$. In all other cases, answers to direct queries to $\mathcal{O}'$ are independent from b (queries on different key values).
- Similarly, the probability to have $r = r_i'$ for some i is bounded by $\frac{l}{2^n}$. In all other cases, answers to queries to $\mathcal{O}$ are independent from b (queries on different tweak values).

As a consequence,

$$\mathsf{Adv}_{\Pi}^{\mathsf{ind-cpa}}(l) \leq \frac{1}{2} + \frac{2l}{2^n}.$$

□

Remarks:

- The proof also holds in the stateful case. The only difference is that the case $r = r_i$ can then never happen, resulting in a slightly better bound, namely $\frac{1}{2} + \frac{l}{2^n}$.
- CPA security obviously assumes that the adversary cannot control the random nonce r used for encryption. By contrast, it is worth noting that the construction of Theorem. 1 did not need to prevent this control of r by the adversary in order to achieve a PRF, and is thus slightly more general.
- Interestingly, we see that the use of a TBC brings the same advantage over block cipher-based constructions (i.e. natural beyond-birthday security) as in the context of authenticated encryption [14].

3.3 Concrete instantiations

Instantiating the previous fresh re-keying schemes essentially requires to specify a block cipher BC, a re-keying function g, and possibly a tweakable block cipher TBC. For the block cipher, a natural choice is the AES. For the re-keying function, since its requirements are the same as in [16], we can stick with their instance and use the following polynomial multiplication in $\mathbb{F}_{2^8}[y]$ modulo $p(y) = y^{16} + 1$:

$$g : (\mathbb{F}_{2^8}[y]/p(y))^2 \to \mathbb{F}_{2^8}[y]/p(y), \quad (k, r) \mapsto k \cdot r.$$

This choice indeed fulfills the additional requirement in Theorem 1 that $\mathsf{g}(K, \cdot), \mathsf{g}(\cdot, R)$ are one-to-one with high probability (the same fact is exploited to invert this function in the CARDIS 2014 collision attack). Note that the analysis of the side-channel behaviour of the polynomial multiplication has only been done for random nonces r by Medwed et al. [16]. Thus, the polynomial multiplication should only be used

in this scenario. Finding instances of g that behave well in the hybrid case is an interesting scope for further research. For the tweakable block cipher, we suggest to use the efficient instances listed in introduction (i.e. Deoxys, Jotlik, KIASU, and Scream). AES-based solutions can also be exploited, by considering the block cipher-based constructions in [14]. Mennink also proposes optimal techniques to build a tweakable block cipher from a block cipher in [17] .

4 Application to a hybrid scheme by Kocher

We now investigate another alternative to hybrid re-keying that was patented by Kocher in [13]. We first recall that this scheme can be compromised using the CARDIS 2014 collision attack. We then describe how to fix it with a tweakable block cipher.

4.1 The CARDIS 2014 attack against Kocher's hybrid re-keying

Description of the scheme. In Kocher's re-keying, and in contrast with the schemes of Section 2.1, the session key is not derived from a static secret master key with the help of randomly generated nonces. Instead, the secret itself is updated, changed, and used as session key. The update is based on the tree structure that is depicted in Fig. 4.

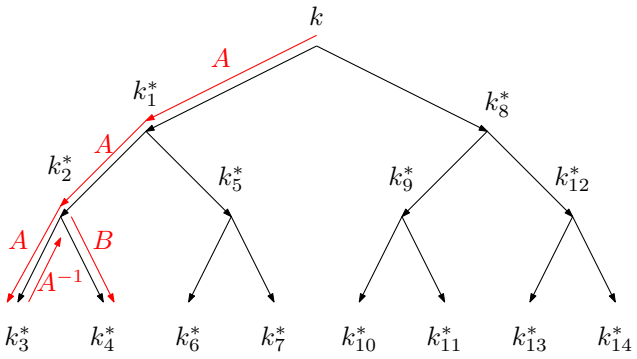


Fig. 4. Hybrid re-keying patented by Kocher [13].

The root of the tree is the secret master key k and the other vertices represent session keys k_i^*. To traverse through the tree, the functions A, B, A^{-1}, and B^{-1} are used, where A^{-1}, and B^{-1} are the inverse functions of A, and B. For instance if we want to go from k to k_1^* in Fig. 4, we calculate $k_1^* = A(k)$. If we want to go from k to k_8^*, we calculate $k_8^* = B(k)$. The number of usable session keys k_i^* is determined by the depth of the tree, which has to be fixed in advance.

We assume for this scheme a similar use-case as supposed for the Africacrypt 2010 scheme. Thus, one party (e.g. the tag) needs easy and cheap protection against side-channel attacks, whereas the other party (e.g. the reader) has to be protected by

other mechanisms. To realize this, the tag strictly follows the tree, which means that it uses the session keys in the strict order given in Fig. 4 ($k_1^*, k_2^*, ..k_n^*$). The tag tells the other party (the reader) the index i of the currently used session key k_i^*, and the reader calculates the session key k_i^* starting from the root k.

Note that other variants of this scheme are possible. For instance, session keys corresponding to internal vertices can be used three times (every time a vertex is visited). By doing so, the number of transitions for the tag can be limited to one. In other words, this reuse of session keys allows the tag to perform only one of the operations A, B, A^{-1}, or B^{-1} per change of the session key.

Collision attack. As already hinted by Dobraunig et al. [9], the CARDIS 2014 collision attack also applies to re-keying schemes like Kocher's one. For simplicity, we make the same assumptions as in Section 2.3 (i.e. each session key is used in a single block cipher execution, and the attacker can choose the plaintext which is encrypted). In addition, we assume that the concrete instance of Kocher's scheme only uses every session key once.

The first step of the CARDIS 2014 attack described in Section 2.3 is to recover one session key: this step goes exactly as before, i.e. building (offline) a database of encryptions of the same plaintext with various keys, then relying on the birthday paradox to obtain collisions with (online) encryptions of the same plaintext. Next, and as far as the second step is concerned, if we additionally assume that the used operations (permutations) A, B, A^{-1}, and B^{-1} are publicly known, one recovered session key k_i^* and the corresponding index i are enough to recover the master key k. Note that keeping the A, B, A^{-1}, and B^{-1} permutations secret would typically mean implementing them as a block cipher with secret (fixed) key, which would then be a potential target to DPA (i.e. lead to a chicken and egg problem, essentially).

4.2 Efficient fix based on a tweakable block cipher

To get rid of the CARDIS 2014 attack, we propose a fix for Kocher's scheme based on a tweakable block cipher similar to Section 3.2. Here, the session key k_i^* – generated by Kocher's scheme (Fig. 4) – is used as a secret key for the tweakable block cipher, and the index i of the key is used as tweak, as illustrated in Fig. 5. The collision attack does not work in this case, for the same reasons as described in Section 3.2 (namely, different tweaks, and so virtually different block ciphers, are used with different keys).

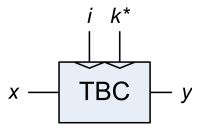


Fig. 5. Hybrid re–keying with a tweakable block cipher.

5 Encryption vs. authentication issues

Our discussion so far dealt with the generic problem of designing a secure and side-channel resistant re-keying scheme. Since they originally borrow from Abdalla and Bellare, our solutions typically lead to secure encryption per se. However, it is worth mentioning that additional security issues might arise when dealing with other uses of this re-keying scheme. In particular, although we ruled out key recovery attacks by ensuring that the master key cannot be recovered from a session key, it should be pointed out that, in some contexts such as authentication, even the recovery of one single session key might already be a serious threat.

Consider a simple challenge–response protocol carried out between two parties. In this scenario we have a tag that acts as prover and a reader that acts as verifier. So in the first step, the reader sends the challenge x_i to the tag. The tag encrypts the challenge and responds with $y_i = E_{k_i^*}(x_i)$. We assume that a re-keying scheme is used, which changes the key k_i^* for every new call of the encryption. As before, one of the n-bit session keys k^* can be recovered with a complexity of about $2 \cdot 2^{n/2}$, regardless of the re-keying scheme actually used.

Now examine the implications of one session key recovery in this scenario, for actual re-keying schemes. If the Africacrypt 2010 scheme is used to generate the session key k^*, the tag (prover) decides alone of the nonce value r. So an attacker does not necessarily need the master key. It is already enough to have one session key to pass the challenge–response protocol, since the attacker can force the use of a single recovered session key. Concretely, the attacker would first select a plaintext value X and build off-line a DB of encryptions of X with random keys. He would then play the role of a reader and query a genuine tag with challenge X. Finally, having recovered one session key, he would be able to impersonate the tag by always using r as nonce. The same attack would also succeed against the tree-based session key generation scheme of Section 4.1.

This attack cannot be prevented by making g not invertible. In fact, this attack always applies to every re-keying scheme, as long as a block cipher is re-keyed for every encryption and the prover can determine the session key to be used. This means that this attack might be also applicable to schemes where the CARDIS 2014 attack does not work. Belaïd et al. [2] presented such a scheme, where the re-keying function g of the Africacrypt 2010 scheme is replaced by a non-invertible function. However, the applicability of the session key replay attack depends on the actual method to determine the nonce, which is not specified by Belaïd et al.

In general, there exist two countermeasures against such a session key replay attack. The first one is to have both parties contributing to the selection process of the session key, as is already done in the CARDIS 2011 scheme [15]. The other one is to prohibit the recovery of the session key in the first place. Interestingly, the constructions using tweakable block ciphers we propose in Sections 3.2 and 4.2 do prevent this recovery. Indeed, as discussed in Section 3.2, changing the value of r will not only result in a different session key k^*, but also in a different tweak value and hence – assuming a perfect tweakable block cipher – virtually in a different block cipher. As a consequence, the collision attack to recover session keys is not applicable any more in this case.

6 Conclusion: side-channel security

We conclude this paper with a cautionary note regarding the exact security improvements brought by fresh re-keying regarding side-channel attacks. For this purpose, the first positive observation is that if the adversary targets the re-keying function and the block cipher independently, the resulting security guarantees are well understood. That is, the implementation will be secure as long as g resists DPA and BC (or TBC) resists SPA. But quite naturally, security arguments and proofs also indicate what are the potential weak points of a construction, and this is clearly the case for fresh re-keying. Looking at Fig. 1, 2a and 2b, this potential weak point is indeed the interaction between the re-keying function and the block cipher. That is, if some leakage about k^* (in Fig. 1) or w^* (in Fig. 2a, 2b) is obtained by the adversary (e.g. when recombining the shares after the masked execution of g), attacks combining mathematical cryptanalysis and leakage (e.g. the algebraic SPA described in [15]) are likely to be very powerful to accumulate partial information on the master key k and finally recover it. This is why fresh re-keying crucially relies on the SPA security of the block cipher, and a secure implementation should recombine the shares of the fresh keys in a sufficiently secure, i.e. typically shuffled, manner (as clearly mentioned in [15] as well). Note that this observation does not annihilate the interest of fresh re-keying which still significantly reduces the adversary's attack paths (compared to the straightforward execution of a block cipher). Interestingly, a very similar situation can be found for the SPRING primitive discussed in [7], which also aims at an informal separation between the parts of the primitive that are easy to mask, and those that are not (and therefore need to be carefully shuffled). Besides, such an issue does not directly apply to Kocher's hybrid scheme which does not make use of a g function. Indeed, combining small leakage on the session keys would require to go through the permutations A and B (and their inverses), which may not be easy if they are implemented with fixed key block ciphers. But this comes at the cost of a slightly more expensive key update mechanism. Besides, and more importantly, it makes any attempt to secure both the encrypting/proving and the decrypting/verifying parties of a protocol much more challenging, since the tree-based construction in Fig. 4 has to be stateless. In other words, it does not benefit from the malleability of the g function which is exploited for this purpose in multi-parties fresh re-keying [15], which gives a typical application of the "no free lunch" theorem.

Acknowledgments. This work has been supported in part by the Austrian Science Fund (project P26494-N15), by the Austrian Government through the research program ICT of the Future under the project number 4593209 (project SCALAS), by the Brussels Region Research Funding Agency through the program Secur'IT and by the European Commission through the ERC project 280141 (CRASH) and the COST Action CRYPTACUS. F.-X. Standaert is a research associate of the Belgian Fund for Scientific Research (FNRS-F.R.S.).

References

1. Abdalla, M., Bellare, M.: Increasing the lifetime of a key: A comparative analysis of the security of re-keying techniques. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 546–559. Springer (2000)
2. Belaïd, S., De Santis, F., Heyszl, J., Mangard, S., Medwed, M., Schmidt, J., Standaert, F., Tillich, S.: Towards fresh re-keying with leakage-resilient PRFs: cipher design principles and analysis. J. Cryptographic Engineering 4(3), 157–171 (2014)
3. Belaïd, S., Grosso, V., Standaert, F.: Masking and leakage-resilient primitives: One, the other(s) or both? Cryptography and Communications 7(1), 163–184 (2015)
4. Bellare, M., Kohno, T.: A theoretical treatment of related-key attacks: RKA-PRPs, RKA-PRFs, and applications. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 491–506. Springer (2003)
5. Biham, E., Shamir, A.: Differential fault analysis of secret key cryptosystems. In: Kaliski Jr., B.S. (ed.) CRYPTO '97. LNCS, vol. 1294, pp. 513–525. Springer (1997)
6. Black, J., Rogaway, P., Shrimpton, T., Stam, M.: An analysis of the blockcipher-based hash functions from PGV. J. Cryptology 23(4), 519–545 (2010)
7. Brenner, H., Gaspar, L., Leurent, G., Rosen, A., Standaert, F.: FPGA implementations of SPRING - and their countermeasures against side-channel attacks. In: Batina, L., Robshaw, M. (eds.) CHES 2014. LNCS, vol. 8731, pp. 414–432. Springer (2014)
8. CAESAR Competition: http://competitions.cr.yp.to/caesar-submissions.html
9. Dobraunig, C., Eichlseder, M., Mangard, S., Mendel, F.: On the security of fresh re-keying to counteract side-channel and fault attacks. In: Joye, M., Moradi, A. (eds.) CARDIS 2014. LNCS, vol. 8968, pp. 233–244. Springer (2014)
10. Dziembowski, S., Pietrzak, K.: Leakage-resilient cryptography. In: FOCS 2008. pp. 293–302. IEEE Computer Society (2008)
11. Grosso, V., Standaert, F., Faust, S.: Masking vs. multiparty computation: how large is the gap for AES? J. Cryptographic Engineering 4(1), 47–57 (2014)
12. Jean, J., Nikolic, I., Peyrin, T.: Tweaks and keys for block ciphers: The TWEAKEY framework. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014. LNCS, vol. 8874, pp. 274–288. Springer (2014)
13. Kocher, P.C.: Leak-resistant cryptographic indexed key update (2003), US Patent 6,539,092
14. Liskov, M., Rivest, R.L., Wagner, D.: Tweakable block ciphers. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 31–46. Springer (2002)
15. Medwed, M., Petit, C., Regazzoni, F., Renauld, M., Standaert, F.: Fresh re-keying II: securing multiple parties against side-channel and fault attacks. In: Prouff, E. (ed.) CARDIS 2011. LNCS, vol. 7079, pp. 115–132. Springer (2011)
16. Medwed, M., Standaert, F., Großschädl, J., Regazzoni, F.: Fresh re-keying: Security against side-channel and fault attacks for low-cost devices. In: Bernstein, D.J., Lange, T. (eds.) AFRICACRYPT 2010. LNCS, vol. 6055, pp. 279–296. Springer (2010)
17. Mennink, B.: Optimally secure tweakable blockciphers. In: Leander, G. (ed.) Fast Software Encryption - 22nd International Workshop, FSE 2015, Istanbul, Turkey, March 8-11, 2015, Revised Selected Papers. Lecture Notes in Computer Science, vol. 9054, pp. 428–448. Springer (2015), http://dx.doi.org/10.1007/978-3-662-48116-5_21
18. Yu, Y., Standaert, F., Pereira, O., Yung, M.: Practical leakage-resilient pseudorandom generators. In: Al-Shaer, E., Keromytis, A.D., Shmatikov, V. (eds.) CCS 2010. pp. 141–151. ACM (2010)